Computational Intelligence and Deep Learning Project Report

Convolutional Neural Network for Medical Imaging Analysis - Abnormality Detection in Mammography

Mirco Quintavalla

Academic Year 2020/2021

Contents

1 Task 1: Analysis of the Most Relevant Works **4**
- 1.1 Tsochatzidis et al. : a Comparative Study 4
- 1.2 Xi et al. : Calcification-Mass Classification and Localization . . . 7
- 1.3 Shen et al. : from Patch to Full Image Classifier 9

2 Task 2: Training from Scratch **14**
- 2.1 Subtask 1: Calcification-Mass Classification 14
  - 2.1.1 Experiment 1_1: a Quick Test to Find a Promising Model 14
  - 2.1.2 Experiment 1_2: Early Stopping, Dropout, Data Augmentation 19
  - 2.1.3 Experiment 1_3: Hyperparameters tuning and other optimizations 20
  - 2.1.4 Conclusions 21
- 2.2 Subtask 2: Benign-Malignant Classification 21
  - 2.2.1 Experiment 2_1: usage of Previously-Created Networks . 21
  - 2.2.2 Experiment 2_2: Early Stopping, Dropout, Data Augmentation 24
  - 2.2.3 Experiment 2_3: Hyperparameters tuning and other optimizations 24
  - 2.2.4 Conclusions 25

3 Task 3: Usage of a Pre-Trained Network **26**
- 3.1 Subtask 1: Calcification-Mass Classification 26
  - 3.1.1 Experiment 1_1: Feature Extraction from VGG16 and ResNet50 26
  - 3.1.2 Experiment 1_2: Extract Only First Layers from VGG16 29
  - 3.1.3 Conclusions 30
- 3.2 Subtask 2: Benign-Malignant Classification 30
  - 3.2.1 Experiment 2_1: Feature Extraction from VGG16 and ResNet50 30
  - 3.2.2 Experiment 2_2: Extract Only First Layers from VGG16 33
  - 3.2.3 Conclusions 34

4 Task 4: Usage of Baseline Patches **35**
- 4.1 Subtask 1: Benign-Malignant Classification 35
  - 4.1.1 Experiment 1: Feeding Models with Baselines for Benign-Malignant Classification - from-Scratch models 35
  - 4.1.2 Experiment 2: Feeding Models with Baselines for Benign-Malignant Classification - pre-trained models 39
  - 4.1.3 Conclusions 41

5 Task 5: Developement of a Composite Classifier **42**
- 5.1 Subtask 1: Calcification-Mass Classification 42
  - 5.1.1 Experiment 1_1: Majority Voting 42

5.1.2 Experiment 1_2: Average Voting 43
5.1.3 Experiment 1_3: Weighted Average Voting 43
5.1.4 Conclusions 44
5.2 Subtask 2: Benign-Malignant Classification 44
5.2.1 Experiment 2_1: Majority Voting 45
5.2.2 Experiment 2_2: Average Voting 45
5.2.3 Experiment 2_3: Weighted Average Voting 46
5.2.4 Conclusions 47

References **48**

1 Task 1: Analysis of the Most Relevant Works

Different solutions to this problem have already been studied and proposed. The aim of this section is to collect the most important ones and briefly discuss and analyze the adopted methodology and the achieved results.

1.1 Tsochatzidis et al. : a Comparative Study

[TCP19] analyses a set of possible solutions based on different networks. In particular, two scenarios are investigated: pre-trained weights and randomly initialized weights. The classification performed is in the context of breast mass, and the classes are *bening* and *malignant*. Input images are assumed to be 224x224 sized.

The considered Convolutional Neural Networks are the following.

- **AlexNet**. This network is made up by five convolutional layers and three fully connected layers. The overall structure is represented in Figure 1.

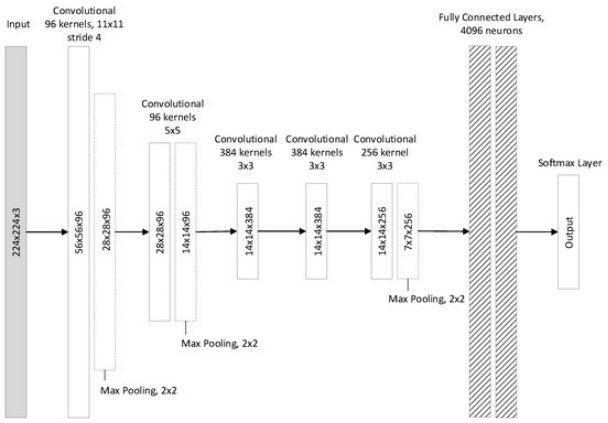

Figure 1: AlexNet Architecture

The number of neurons in the output layer is equal to the number of classes. A *ReLu* function can be used as activation function instead of a *Softmax* one, so that vanishing gradient problem is mitigated. The output of each convolutional layer can be normalized using a depth-wise normalization scheme called *Local Response Normalization* (LRN). For the three fully connected layers dropout (with a probability of 0.5) is used in order to avoid overfitting.

- **VGG** *VGG* is a family of networks that uses very small receptive fields (3x3 with a stride of 1). The small-sized convolution filters allows VGG

to have a large number of weight layers, which leads to improved performance. A *ReLu* activation function is used in the convolutional layers. Similarly to *AlexNet*, a stack of fully connected layers is put on the convolutional part of the network. Multiple configurations of total numbers of layers are possible, however best results are reached by depths between 16 and 19 layers. The structure of *VGG16* is represented in Figure 2.

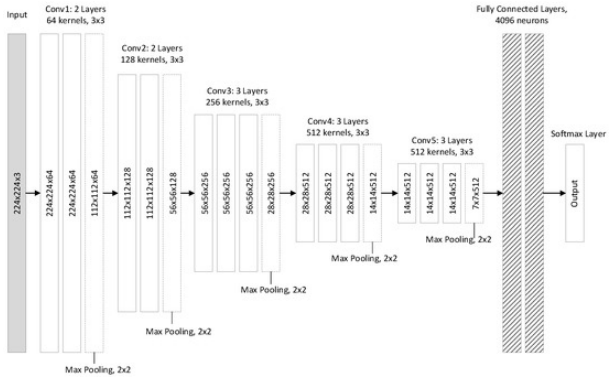

Figure 2: VGG16 Structure

- **GoogLeNet** *GoogLeNet* networks are based on *Inception* module. The idea is to find to optimal local structure and repeat it. The same input is given to four different branches, and the result of each one is finally concatenated with the other results. Since the middle layers of a CNN can produce discriminative features, classifiers are added to operate on these features. The computed loss is used in back-propagation step.

- **ResNet** *ResNet* is a family of networks that learn residual functions with reference to the inputs. For a small set of convolutional layers, a shortcut (implementing identity mapping) is added in parallel to these layers and a sum of the results of the two branches is performed. Small kernels (3x3 sized) are used. Since these networks are easier to optimize, the total number of layers can be significantly increased (even up to 150 layers).

The **dataset** considered is CBIS-DDSM. Since the classification regards cases concerning mass only, only a subset of whole dataset is considered. In conclusion, training set and test set are made up by 1319 and 378 patches respectively. Data augmentation is performed in order to avoid overfitting.

Talking about the **training** strategy, *Glorot* initialization is used for training from scratch. This method initializes each weight with a small Gaussian value with $mean = 0.0$ and *variance* based on the fan-in and fan-out of the weight. On the other hand, for fine-tuning experiments weights are initialized with the ones obtained from *ImageNet* training. Fully connected layers of pre-trained

networks are removed and replaced with randomly initialized ones. The final layer of each network is fully connected and contains two *Softmax* neurons for the binary classification. Training step is early stopped if AUC on validation set does not improve for 15 consecutive epochs. Training parameters are summed up in Table 1.

Hyperparameters used in Training Step				
CNN	Batch Size	Learning Rate	Optimizer	Number of Epochs
AlexNet	32	1e-6 (FT and SC)	Adam	/
VGG16	32	1e-6 (FT and SC)	Adam	/
VGG19	32	1e-6 (FT and SC)	Adam	/
ResNet50	32	1e-6 (FT), 1e-5 (SC)	Adam	/
ResNet101	16	1e-6 (FT and SC)	Adam	/
ResNet152	10	1e-6 (FT), 1e-8 (SC)	Adam	/
GoogLeNet	32	1e-6 (FT and SC)	Adam	/

Table 1: Hyperparameters used in Training Step (FT = fine-tuning, SC = from scratch)

Talking about the **results**, networks trained in fine-tuning scenario perform better than the ones trained from scratch. Best performance is achieved by fine-tuning *ResNet50* and *ResNet101*. *ResNet* performances are low in from-scratch training scenario, probably because the high number of layers makes the convergence more difficult and increases the probability of overfitting. Talking about from-scratch training, the best performance is achieved by *AlexNet*, which is the simplest network. Results are summed up in Table 2.

Results of NNs				
CNN	AUC (SC)	ACC (SC)	AUC (FT)	ACC (FT)
AlexNet	0.716	0.656	0.802	0.753
VGG16	0.702	0.580	0.781	0.716
VGG19	0.707	0.581	0.781	0.716
ResNet50	0.637	0.627	0.804	0.749
ResNet101	0.641	0.662	0.791	0.753
ResNet152	0.609	0.647	0.793	0.755
GoogLeNet	0.590	0.598	0.767	0.720

Table 2: Performances of Neural Networks (AUC = Area Under the Curve, ACC = accuracy, FT = fine-tuning, SC = from scratch)

1.2 Xi et al. : Calcification-Mass Classification and Localization

A possible solution to the problem is the one proposed in [XSG18]. The primary idea is to exploit pre-trained CNNs on image patches of calcification and masses, fine-tune them and using full images as input of the network in order to classify between the two classes (calcification and mass). Finally abnormalities are localized by means of the computation of *Class Activation Maps* (CAM).

The problem of computer-aided mammography cannot be be treated as a traditional image classification task because abnormalities represent a small fraction of the full image (typically, in a 3000x4600 sized mammography the abnormality region is only 200x200 pixels). For that reason the solution proposed in [XSG18] trains deep CNNs on image patches and then adapts them to full images. Firstly, a binary classifier is trained with deep CNN architectures using transfer learning. Output layers are modified to have two outputs classes and they are fine-tuned while the first part of the newtork is frozen. Finally, abnormalities are classified and detected in full images. The overall architecture is described in Figure 3.

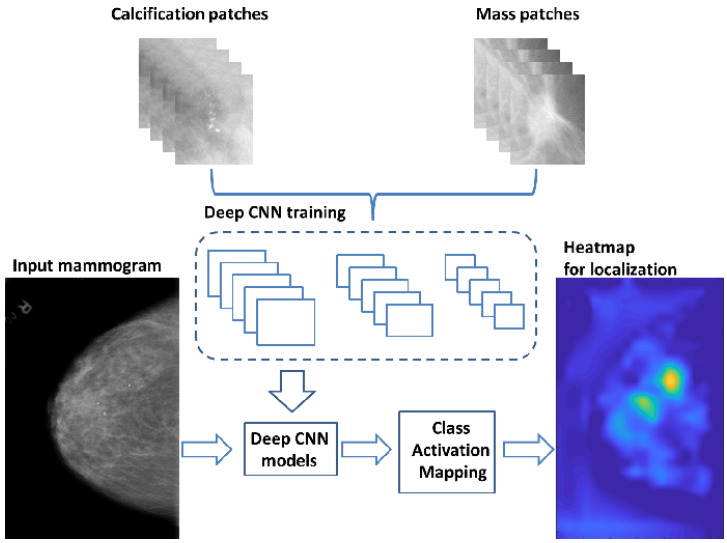

Figure 3: General Overview of Xi approach [XSG18]

The **dataset** used is the CBIS-DDSM, which contains 753 calcification cases and 891 mass cases. CBIS-DDSM training and test sets are merged and then splitted again using a 85-15 proportion. In conclusion, 1511 calcification (1284 for training and 227 for testing) and 1592 mass (1353 for training and 239 for testing) patches are used. To avoid overfitting, data augmentation is performed on training data (random rotation between 0 and 360 degrees, random X and

Y reflections.

Talking about the different deep CNNs used, the solution proposed in [XSG18] adapts four models to the problem of classifying calcification and mass in mammograms: *AlexNet*, *VGG*, *GoogLeNet* and *ResNet*. The last three layers are removed from each network and subsituted with a fully connected layer, a *Softmax* layer and a classification layer. In order to train and test each architecture, five-fold cross validation is used.

To identify **abnormalities position**, *Class Activation Map* (CAM) is computed. A deep CNN is cut after the last convolutional layer and a global average pooling layer and a fully connected layer are appended. Consequently the model needs to be retrained to learn output layer weights. *ResNet* has already this architecture so it is used to compute CAM. In conclusion, the importance of each region can be computed as:

$$CAM = \sum_{i=1}^{n} w_i f_i$$

where w represents the weights related to the output layer and f represents the feature maps from the output of the last convolutional layer.

Talking about **training** strategy, the most important choices are summed up in Table 3.

Hyperparamters used for Model Training	
Hyperparameter	Value
Max Number of Epochs	200
Optimizer	SGDM (Stochastic Gradient Descent with Momentum)
Batch Size	16
Learning Rate	Initial 1e-4; learning rate factor for the last fully connected layer 20.0
Stopping Condition	Mean accuracy on fifty most recent batch reaches 99.5% or Max Number of Epochs is reached

Table 3: Hyperparameters used for network training

The **results** are evaluated by overall accuracy. Table 4 shows the accuracies obtained by using different pre-trained models.

Mean Classification Accuracies of Deep CNNs	
Model	Overall Accuracy
VGGNet	92.53%
ResNet	91.80%
AlexNet	91.23%
GoogleNet	91.10%

Table 4: Mean Classification Accuracies of Deep CNNs

Finally, Figure 4 represents an example of mass localization result.

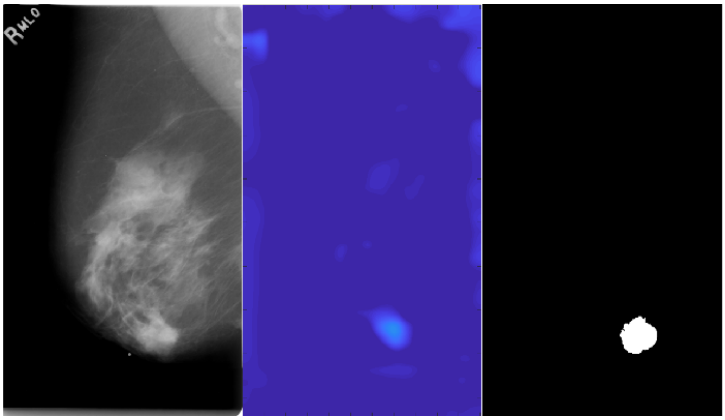

Figure 4: Mass localization result (full mammogram, class activation map output, ground-truth binary mask image)

1.3 Shen et al. : from Patch to Full Image Classifier

Another similar possible way to solve the problem is described in [She+19]. This solution employs a two-phase approach: the model is firstly pre-trained to classify local image patches using a fully annotated dataset with Region Of Interest (ROI) information, and then the resulting weights are used to initialize a full image classifier, which is then fine-tuned (even with a non-annotated dataset).

Formally, given an input patch $X \in R^{p \times q}$, the patch classifier can be seen as a function f such that $f(X) \in R^c$, where c is the number of classes the classifier is able to distinguish. The output of f represents a vector containing the probabilities the patch X belongs to each class, so each value is in $[0, 1]$. Given an image $M \in R^{r \times s}$ with $p \ll r$ and $q \ll s$, f (that is a CNN) can be applied without changing parameters so that $f(M) \in R^{u \times v \times c}$ with $u > 1$ and $v > 1$. In other words, the output represents an heatmap of probabilities. Layers (using a convolutional one for latest for the purpose of creating an "all convolutional" network) are then added on top of heatmap in order to obtain the

definitive classification. In summary, considering g as the function representing the top layers, the whole image classifier can be seen as $h(M) = f(g(M)) \in R^d$, where d is the number of classes we finally want to predict (usually $d = 2$, *benign* and *malignant*). It can be noticed that the heatmap creates a bottleneck-like structure which can led to information loss, so it is useful to remove it in whole image classifier. To sum up, the general structure can be represented as in Figure 5.

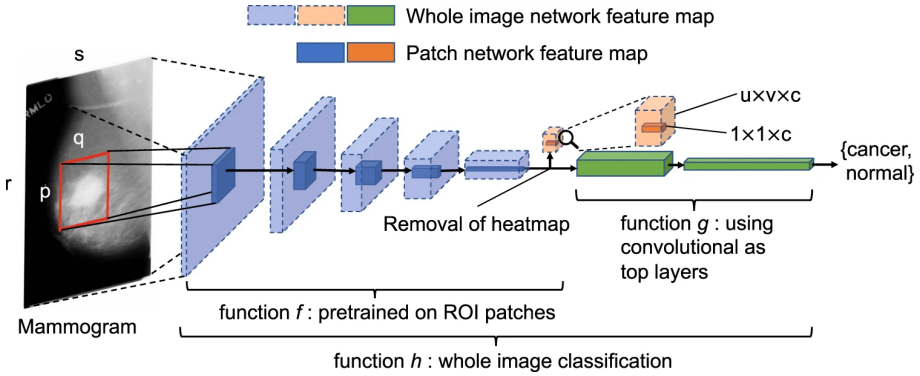

Figure 5: General Overview of Shen approach [She+19]

Talking about the specific **network design**, the models presented in [She+19] makes uses of two CNN structures, *VGG16* and Residual Networks (*ResNet50*). These models are used as patch classifiers. To be consistent with *Resnet50*, the two fully connected layers of *VGG16* are replaced with a global average pooling layers that calculates the average activation of each feature map of the last *VGG16* block (a block is a group of consecutive layers).
In order to pass from a patch classifier to a full-image classifier, the heatmap is flattened and fully connected layers are added on the top of the patch classifier. *ReLu* is used as patch classifier activation function in order to avoid vanishing gradient problem. Different experiments, such as removing the heatmap and adding convolutional layers on the top of patch classifier, are then performed to achieve the best result.

The **dataset** used is CBIS-DDSM, which contains 2478 mammography images (from 1249 patients) in DICOM format. The final classification is performed to evaluate the benign or malignant condition of each image. The split percentage used is 85-15, meaning that 85% of the dataset is considered as training set and the remaining 15% composes the test set. The validation set is formed by isolating the 10% of the patients of the training set. Proportion of malignant cases is mantained constant between the three sets. To sum up, training, test and validation sets are composed by 1903, 199 and 376 images respectively.
Each mammography is converted in PNG format and downscaled to 1152 $\times$

896. Two patch image sets are then created by sampling patches from ROIs and background region respectively. The size of each patch is 224 × 224. The first set (called S1) is composed by pairs of patches in which the first one is centered on the ROI and the other one is a random background patch from the same image. The second set (called S10) is obtained considering 10 patches around each ROI (with a minimum overlapping ratio of 0.9) and the same number of background patches from the same image. Patches are finally categorized into five classes: background, malignant mass, benign mass, malignant calcification and benign calcification.

Another important aspect is the **training** approach. As a matter of fact, the first step is to initialize weights for the patch classifier. This can be done by randomly initializing them or using pre-trained weights, using *ImageNet* for example. Patch classifier training methodologies are described in Table 5, 6, 7 and 8. They make use of a 3-stage strategy in order to avoid to lose all the features learned in bottom layers.

Set S1 and Randomly Initialized Weights	
Hyperparameter	Value
Number of Epochs	200
Optimizer	Adam
Batch Size	32
Learning Rate	STAGE 1: 1e-3 to train the last layer STAGE 2: 1e-3 unfreezing the top 46 layers for ResNet50 and top 11 for VGG16 STAGE 3: 1e-3 unfreezing all layers

Table 5: Hyperparameters used for patch classifier training (S1 and Randomly Initialized Weights)

Set S10 and Randomly Initialized Weights	
Hyperparameter	Value
Number of Epochs	50
Optimizer	Adam
Batch Size	32
Learning Rate	STAGE 1: 1e-3 for 3 epochs to train the last layer STAGE 2: 1e-3 for 10 epochs, unfreezing the top 46 layers for ResNet50 and the top 11 for VGG16 STAGE 3: 1e-3 for 37 epochs, unfreezing all layers

Table 6: Hyperparameters used for patch classifier training (S10 and Randomly Initialized Weights)

Set S1 and Pre-Trained Weights	
Hyperparameter	Value
Number of Epochs	200
Optimizer	Adam
Batch Size	32
Learning Rate	STAGE 1: 1e-3 to train the last layer STAGE 2: 1e-4 unfreezing the top 46 layers for ResNet50 and top 11 for VGG16 STAGE 3: 1e-5 unfreezing all layers

Table 7: Hyperparameters used for patch classifier training (S1 and Pre-Trained Weights)

Set S10 and Pre-Trained Weights	
Hyperparameter	Value
Number of Epochs	50
Optimizer	Adam
Batch Size	32
Learning Rate	STAGE: 1e-3 for 3 epochs to train the last layer STAGE 2: 1e-4 for 10 epochs, unfreezing the top 46 layers for ResNet50 and the top 11 for VGG16 STAGE 3: 1e-5 for 37 epochs, unfreezing all layers

Table 8: Hyperparameters used for patch classifier training (S10 and Pre-Trained Weights)

After the training of the patch classifier the full image classifier can be trained. Firstly learning rate and weight decay are set to 1e-4 and 1e-3 respectively to train new top layers for 30 epochs. Then the same hyperparamters are changed to 1e-5 and 1e-2 respectively to train all layers for 20 epochs. Batch size is reduced to 2 in full-image classifier training step. During training step, the average gray scale value of the whole image training set is subtracted from both patch and full-image sets.

Talking about the **results**, the accuracy of the patch classifier on the test set varies from 0.63 (*Resnet50*, not pre-trained, S10) to 0.99 (*Resnet50*, pre-trained, S1). For the full image classifier, results are evaluated on the base of per-image AUCs on the test set. Best results are summarized in Table 9.

Best Full Image Classifier Results				
Patch Classifier	Patch Set	Block1	Block2	AUC
ResNet50	S10	256x1	128x1	0.87
ResNet50	S10	[512-512-1024]x2	[512-512-1024]x2	0.85
VGG16	S10	256x1	128x1	0.85
VGG16	S10	[512-512-1024]x2	[512-512-1024]x2	0.85

Table 9: Best results achieved from the full image classifier

Block1 and *Block2* refer to the blocks used on the top of patch classifier. A *VGG16* block is represented by the pattern $N \times K$, where N is the depth of each convolutional layer and K is the number of convolutional layers. A *ResNet50* block is represented by the pattern $[L - M - N] \times K$, where L, M, N represent the depths of the three convolutional layers in a unit and K is the number of units. As shown, some of the best results are obtained using both *ResNet* and *VGG* architectures. Alernative heatmap design configurations are tested, but none of them reaches a good result (with regard to the already shown results). As conclusion, it is important to specify that *VGG16* is a network much smaller than *ResNet50* (15 millions weights parameters against 24 millions) and for that reason it is desiderable when the resources are limited.

2 Task 2: Training from Scratch

The aim of this section is to design and develop an ad-hoc architecture (or architectures) to distinguish mammograms in Calcification or Mass cases (2.1) and in Benign or Malignant cases (2.2).

2.1 Subtask 1: Calcification-Mass Classification

This subsection is related to Calcification-Mass classification. The section is organized as follows. In the first part, three models are created and evaluated on the basis of a predetermined set of Hyperparameters. The best model is selected and other experiments are done (for example, Early Stopping, Dropout and Data Augmentation in case of overfitting). Finally, further optimizations are performed (e.g. Hyperparameters optimization).
In order to do so, labels need to be redefined. At the current state, Calcification cases are characterized by labels 3 and 4, while Mass cases are characterized by labels 1 and 2. Label 0 is assigned to Calcification class and label 1 is assigned to Mass class. It is also important to analyze the balancing of the two classes so that a countermeasure to a possible unbalanced class problem can be taken. Training set class distribution is the following: 54.48% Calcification cases and 45.52% Mass cases. On the other hand, Calcification samples represent the 46.73% of the test set meaning that Mass samples are the 53.27%. Differences are negligible, so classes can be considered equally balanced.

2.1.1 Experiment 1_1: a Quick Test to Find a Promising Model

The aim of this experiment is to find a promising model, namely a model whose accuracy and loss on the test set are satisfying. The experiment is perfomed without Data Augmentation, Early Stopping and Regularization and with a fixed set of Hyperparameters ($batch\_size = 32$, $number\_of\_epochs = 50$, $optimizer = Adam$, $learning\_rate = 1e-4$) derived from laboratory experience and the abovementioned papers. *Binary Crossentropy* is used as loss function. $Validation\_split$ is set to 0.15, according to the value used in [XSG18] and [She+19].

Preprocessing Training and test images are normalized by dividing them for $MAX\_UINT16 = 65535$. Training images (and their respective labels) are shuffled because the parameter $validation\_split$ of $model.fit$ function will take the last portion of the dataset as validation set. To the current state, images (and their labels) are ordered by class (the elements labelled as 0 are at the end of the set). *Permutation* is used to apply the same reordering to both training images and labels. It is also important to check to composition of the validation set: 54.86% of the samples are Calcification ones, and 45.14% are Mass ones.

Model 1_1_A The model is made up by 5 Convolutional Layers, 3 Max Pooling Layers, a Flatten Layer and 3 Dense Layers (the last one performs the

classification step). The overall structure is inspired by *AlexNet* (described in [KSH12]). However, the total number of weights is drastically reduced (from 62M to 1M), among other things because of the lower number of filters and kernel sizes used. As a consequence, the model is easier to train. It is important to notice that the parameter $padding = same$ (combined with $stride = 1$) of Convolutional layers has the effect to mantain input shape in output. Talking about activation functions, *ReLu* is used in convolutional layers in order to avoid Vanishing Gradient Problem. *Sigmoid* is used as activation function in the last layer. The overall architecture is summarized in Figure 6.

```
Model: "MODEL1_1_A"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
CONV_1 (Conv2D)              (None, 48, 48, 32)        2624
_________________________________________________________________
MAX_POOL_1 (MaxPooling2D)    (None, 24, 24, 32)        0
_________________________________________________________________
CONV_2 (Conv2D)              (None, 24, 24, 96)        76896
_________________________________________________________________
MAX_POOL_2 (MaxPooling2D)    (None, 12, 12, 96)        0
_________________________________________________________________
CONV_3 (Conv2D)              (None, 12, 12, 128)       110720
_________________________________________________________________
CONV_4 (Conv2D)              (None, 12, 12, 128)       147584
_________________________________________________________________
CONV_5 (Conv2D)              (None, 12, 12, 96)        110688
_________________________________________________________________
MAX_POOL_3 (MaxPooling2D)    (None, 5, 5, 96)          0
_________________________________________________________________
FLAT_1 (Flatten)             (None, 2400)              0
_________________________________________________________________
DENSE_1 (Dense)              (None, 256)               614656
_________________________________________________________________
DENSE_2 (Dense)              (None, 256)               65792
_________________________________________________________________
DENSE_3 (Dense)              (None, 1)                 257
=================================================================
Total params: 1,129,217
Trainable params: 1,129,217
Non-trainable params: 0
```

Figure 6: Model 1_1_a Architecture

Accuracy and loss on training and validation sets are shown in Figures 7-8.

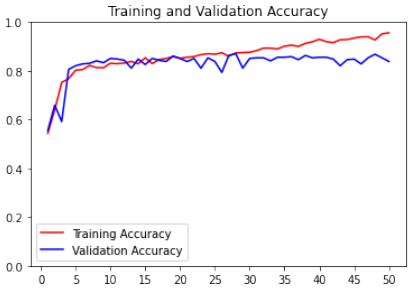

Figure 7: Model 1_1_a Accuracy on Training and Validation Set

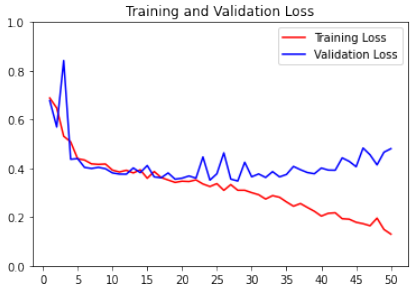

Figure 8: Model 1_1_a Loss on Training and Validation Set

As shown, the model starts to overfit after 20 epochs. As a matter of fact, training accuracy continues to improve while validation accuracy remains constant. It is also confirmed by the fact that validation loss stops decreasing around 20 epochs, while training loss continues falling. Result on the test set are satisfying: 80% of accuracy while loss function reaches 0.62.

Model 1_1_B This model is obtained by stacking four Convolutional layers, each one followed by a Max Pooling layer. There is a single final Fully Connected layer, followed by another one used for the final classification. The usage of more Fully Connected layers led to no improvement. The total number of weights is higher than the ones of Model_1_1_a, and for that reason the training is slower. Once again *ReLu* is used as activation function in Convolutional layers, while *Sigmoid* is used in the last Fully Connected layer. The overall architecture is summarized in Figure 9.

```
Model: "MODEL1_1_B"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
CONV_1 (Conv2D)              (None, 148, 148, 64)      640
_________________________________________________________________
MAX_POOL_1 (MaxPooling2D)    (None, 74, 74, 64)        0
_________________________________________________________________
CONV_2 (Conv2D)              (None, 73, 73, 96)        24672
_________________________________________________________________
MAX_POOL_2 (MaxPooling2D)    (None, 36, 36, 96)        0
_________________________________________________________________
CONV_3 (Conv2D)              (None, 34, 34, 192)       166080
_________________________________________________________________
MAX_POOL_3 (MaxPooling2D)    (None, 17, 17, 192)       0
_________________________________________________________________
CONV_4 (Conv2D)              (None, 15, 15, 192)       331968
_________________________________________________________________
MAX_POOL_4 (MaxPooling2D)    (None, 5, 5, 192)         0
_________________________________________________________________
FLAT_1 (Flatten)             (None, 4800)              0
_________________________________________________________________
DENSE_1 (Dense)              (None, 1024)              4916224
_________________________________________________________________
DENSE_2 (Dense)              (None, 1)                 1025
=================================================================
Total params: 5,440,609
Trainable params: 5,440,609
Non-trainable params: 0
```

Figure 9: Model 1_1_b Architecture

Accuracy and loss on training and validation sets are shown in Figures 10-11.

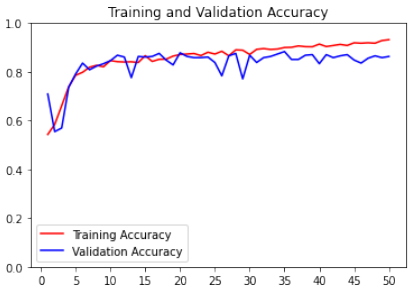

Figure 10: Model 1_1_b Accuracy on Training and Validation Set

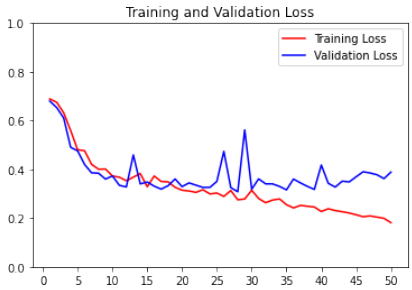

Figure 11: Model 1_1_b Loss on Training and Validation Set

Once again, the model is subjected to overfitting. It stops learning after 20-25 epochs. Plots are very similar to the ones related to Model_1_a, but Validation loss function is more stable. On the other hand, results on the test set are much better: accuracy reaches 85% while loss function 0.41.

Model 1_1_C This model is obtained by stacking more Convolutional blocks, each one composed by two Convolutional layers and a single Max Pooling layer. This idea is similar to the one adopted in *VGG* networks (described in [SZ15]).

Padding = *same* combined with *stride* = 1 ensure that the output of a Convolutional layer has the same shape of the input, so shape reduction is performed only by means of Max Pooling layers. Three Dense layers are finally added to the model, the last one used for the classification. Usage of a lower number of Fully connected layers led to a decrease in performances. Once again, *ReLu* is used as Activation function in Convolutional layers and *Sigmoid* in the last Fully Connected layer. The overall architecture is summarized in Figure 12.

```
Model: "MODEL1_1_C"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
CONV_1 (Conv2D)              (None, 150, 150, 32)      320
_________________________________________________________________
MAX_POOL_1 (MaxPooling2D)    (None, 75, 75, 32)        0
_________________________________________________________________
CONV_2 (Conv2D)              (None, 75, 75, 64)        18496
_________________________________________________________________
MAX_POOL_2 (MaxPooling2D)    (None, 25, 25, 64)        0
_________________________________________________________________
CONV_3 (Conv2D)              (None, 25, 25, 128)       73856
_________________________________________________________________
CONV_4 (Conv2D)              (None, 25, 25, 128)       147584
_________________________________________________________________
MAX_POOL_3 (MaxPooling2D)    (None, 12, 12, 128)       0
_________________________________________________________________
CONV_5 (Conv2D)              (None, 12, 12, 256)       295168
_________________________________________________________________
CONV_6 (Conv2D)              (None, 12, 12, 256)       590080
_________________________________________________________________
MAX_POOL_4 (MaxPooling2D)    (None, 6, 6, 256)         0
_________________________________________________________________
CONV_7 (Conv2D)              (None, 6, 6, 256)         590080
_________________________________________________________________
CONV_8 (Conv2D)              (None, 6, 6, 256)         590080
_________________________________________________________________
MAX_POOL_5 (MaxPooling2D)    (None, 3, 3, 256)         0
_________________________________________________________________
FLAT_1 (Flatten)             (None, 2304)              0
_________________________________________________________________
DENSE_1 (Dense)              (None, 1024)              2360320
_________________________________________________________________
DENSE_2 (Dense)              (None, 1024)              1049600
_________________________________________________________________
DENSE_3 (Dense)              (None, 1)                 1025
=================================================================
Total params: 5,716,609
Trainable params: 5,716,609
Non-trainable params: 0
```

Figure 12: Model 1_1_c Architecture

Accuracy and loss on training and validation sets are shown in Figures 13-14.

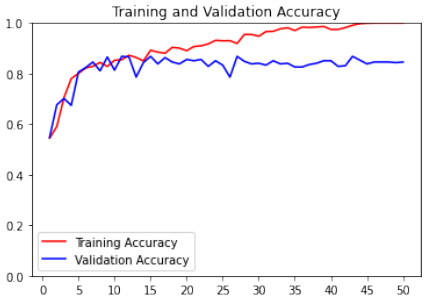

Figure 13: Model 1_1_c Accuracy on Training and Validation Set

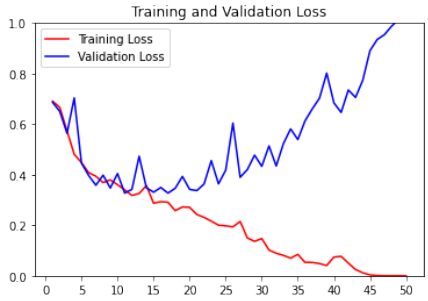

Figure 14: Model 1_1_c Loss on Training and Validation Set

Once again the model is subjected to overfitting. The main difference with previous models is that validation loss function starts increasing after reaching the minimum, while validation accuracy remains constant. This can mean that samples with already bad predictions keep getting worse (for example, the output for a sample of class 1 passes from 0.4 to 0.2). Results on the test set are not as good as Model1_1_b: accuracy is 84%, while loss function is very high (1.13) because of the abovementioned reasons .

Conclusions The three models presents an accuracy on the test set of 80%, 85% and 84% respectively. Model1_1_b seems to be the better because of its high accuracy and low loss function, so it is one considered for next experiments. Model1_1_c presents a good accuracy value, but loss function is much higher than the other models.

2.1.2 Experiment 1_2: Early Stopping, Dropout, Data Augmentation

Since the model is affected by overfitting, an experiment that can be made is using Early Stopping, Dropout and Data Augmentation strategies. This is because overfitting may have decreased the performances on the test test, and these strategies can prevent the model to reach an overfitting situation.

Dropout and Early Stopping A Dropout layer, whose rate is equal to 0.3, is added to the model just before the first Fully Connected layer. In addition, a *Callback* is defined in order to stop the training phase after 5 epochs with no improvement in validation loss function. Weights of the minimum validation loss epoch are then restored.

Accuracy and loss on training and validation sets are shown in Figures 15-16.

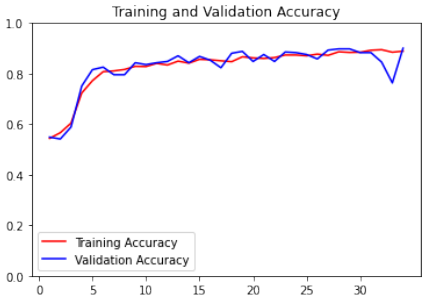

Figure 15: Model 1_1_c Accuracy on Training and Validation Set

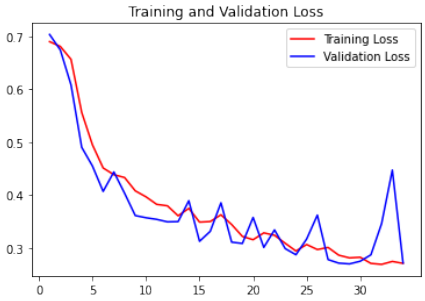

Figure 16: Model 1_1_c Loss on Training and Validation Set

As shown, the model doesn't overfit anymore. Training and validation accuracy follow the same path, and that goes for training and validation loss. Results on the test set are the best ones reached so far: accuracy is 87.5% while loss is 0.34.

Data Augmentation Another experiment that can be done is trying to augment the data. The strategy adopted is the one described in [She+19], in which only rotation (with $range = 90$) is performed. However the experiment produced a little decrease in the performances on the test set. This is probably because data augmentation has the only effect to confuse the training phase. Another reason can be that the samples present no complex structures to be extracted, so data augmentation can barely help.

2.1.3 Experiment 1_3: Hyperparameters tuning and other optimizations

The aim of this experiment is to find possible optimizations of the model by varying the value of the hyperparameters and using particular weights initialization strategies.

A Grid Search can be performed in order to find an optimal configuration of the hyperparameters. Values considered for the grid are derived by papers, articles and laboratory experience, and they are described in the following.

```
optimizers_names = ["Adam", "RMSprop", "SGD"]
learning_rates = [1e-4, 1e-5, 1e-6]
batch_sizes = [16, 32]
dropout_rates = [0.3, 0.4, 0.5]
```

Different experiments are made for different optimizers. *Adam* is used in [TCP19] and [She+19], *SGD* in [XSG18] and *RMSprop* is one of the most used. Learning rates are set to 1e-4, 1e-5 and 1e-6, according to [TCP19] and [XSG18]. Batch size is set to 16 and 32 according to [TCP19], [XSG18] and [She+19]. Finally, Dropout rates are chosen in the set $\{0.3, 0.4, 0.5\}$. Since lower Learning

rates are used, a higher number of epochs (200) and a more patient Callback(15) are used.

In addition, specific kernel initializers are used in accord with TDS Webiste. In particular, *He Uniform* is used for each layer characterized by a *ReLu* activation function, while *Glorot Uniform* is used in those layers in which the activation function is *Sigmoid*.

Conclusions Optimizations didn't led to desired results. Results varies in a small range (0.35-0.5 for loss function, 0.79-0.85 for accuracy) when optimizer is *Adam* or *RMSprop*. On the other hand, results obtained from *SGD* are much worse than the others. This is probably because this optimizer performs better with Pre-Trained networks (see [XSG18]).

2.1.4 Conclusions

The best model is the one developed in 2.1.2, using Callback and Dropout. It reached an accuracy of 87.5% and the loss was 0.34.

2.2 Subtask 2: Benign-Malignant Classification

This subsection is related to Calcification-Mass classification. The section is organized as follows. In the first part, models used in 2.1.1 evaluated for the new classification task on the basis of a predetermined set of Hyperparameters. The best model is selected and other experiments are done (for example, Early Stopping, Dropout and Data Augmentation in case of overfitting). Finally, further optimizations are performed (e.g. Hyperparameters optimization).
In order to do so, labels need to be redefined. At the current state, Benign cases are characterized by labels 1 and 3, while Mass cases are characterized by labels 2 and 4. Label 0 is assigned to Benign class and label 1 is assigned to Malignant class. It is also important to analyze the balancing of the two classes so that a countermeasure to a possible unbalanced class problem can be taken. Training set class distribution is the following: 58.59% Benign cases and 41.41% Malignant cases. On the other hand, Benign samples represent the 65.18% of the test set meaning that Mass samples are the 34.82%. Differences are not so consistent, so classes can be considered equally balanced.

2.2.1 Experiment 2_1: usage of Previously-Created Networks

The aim of this experiment is to find a promising model, namely a model whose accuracy and loss on the test set are satisfying. The experiment is perfomed without Data Augmentation, Early Stopping and Regularization and with a fixed set of Hyperparameters ($batch\_size = 32$, $number\_of\_epochs = 50$, $optimizer = Adam$, $learning\_rate = 1e-4$) derived from laboratory experience and the abovementioned papers. *Binary Crossentropy* is used as loss function. $Validation\_split$ is set to 0.15, according to the value used in [XSG18] and [She+19]. Models used are the ones described in 2.1.1.

Preprocessing Training and test images are normalized by dividing them for $MAX\_UINT16 = 65535$. Training images (and their respective labels) are shuffled because the parameter *validation_split* of $model.fit$ function will take the last portion of the dataset as validation set. Images (and their labels) are shuffled even if they are already randomly distributed. *Permutation* is used to apply the same reordering to both training images and labels. It is also important to check to composition of the validation set: 62.84% of the samples are Benign, and 37.16% are Malignant.

Model 2_1_a Accuracy and loss on training and validation sets are shown in Figures 17-18.

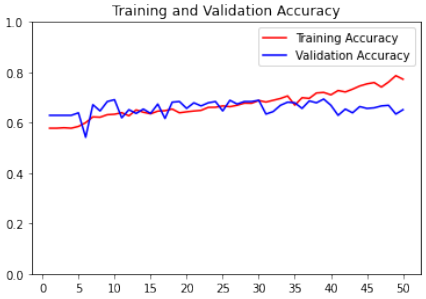

Figure 17: Model 2_1_a Accuracy on Training and Validation Set

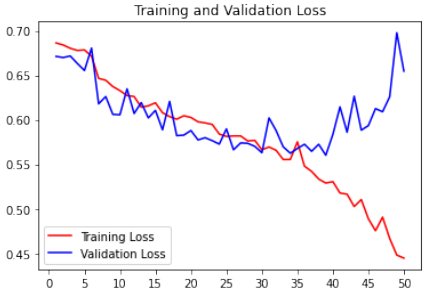

Figure 18: Model 2_1_a Loss on Training and Validation Set

As shown, it is clear that the model overfits after 30-35 epochs. It doesn't seem a case of underfitting because the model is capable to learn, as shown by the increasing training accuracy. At 30-35 epochs, validation loss function starts to increase (probably) because samples with already bad predictions keep getting worse. Accuracy on the test set reaches 60%, and loss function is 0.79. These results are not excellent but they are in line with the ones found in [TCP19].

Model 2_1_b Accuracy and loss on training and validation sets are shown in Figures 19-20.

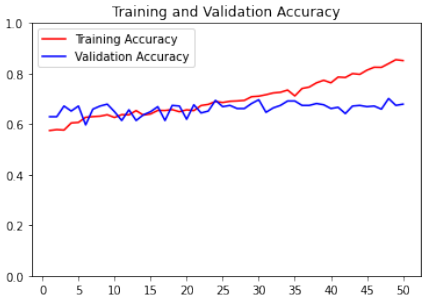

Figure 19: Model 2_1_b Accuracy on Training and Validation Set

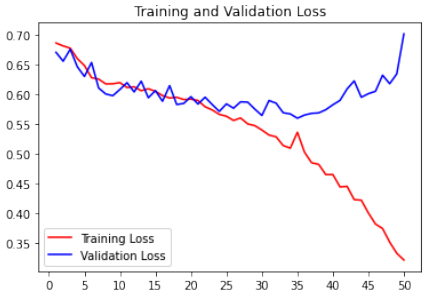

Figure 20: Model 2_1_b Loss on Training and Validation Set

The same considerations made for Model 2_1_a can be made for Model 2_1_b. However, in this case the accuracy on the test set is higher (67%) as well as loss function (0.87). This can mean that the model has a better capacity to identify the class, but wrong-predicted samples have very bad predictions.

Model 2_1_c Accuracy and loss on training and validation sets are shown in Figures 21-22.

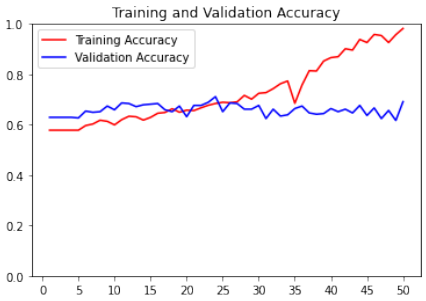

Figure 21: Model 2_1_c Accuracy on Training and Validation Set

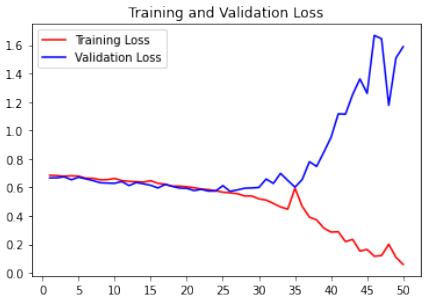

Figure 22: Model 2_1_c Loss on Training and Validation Set

Again, the model overfits after 25 epochs. Accuracy on the test set reaches 58%, while loss function is 1.82. The higher value of the loss function means that the degeneration is quicker than in the previous models.

Conclusions The three models presents an accuracy on the test set of 60%, 67% and 58% respectively. The difference between is not negligible, so the next experiments will consider only the best model so far (Model2_1_b).

2.2.2 Experiment 2_2: Early Stopping, Dropout, Data Augmentation

Since the model is affected by overfitting, an experiment that can be made is using Early Stopping, Dropout and Data Augmentation strategies. This is because overfitting may have decreased the performances on the test test, and these strategies can prevent the model to reach an overfitting situation.

Dropout and Early Stopping A Dropout layer, whose rate is equal to 0.3, is added to the model just before the first Fully Connected layer. In addition, a *Callback* is defined in order to stop the training phase after 5 epochs with no improvement in validation loss function. Weights of the minimum validation loss epoch are then restored.

Accuracy and loss on training and validation sets are shown in Figures 23-24.

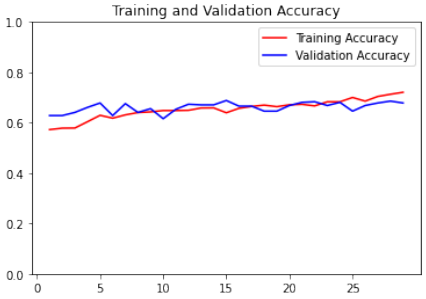

Figure 23: Model 2_2_de Accuracy on Training and Validation Set

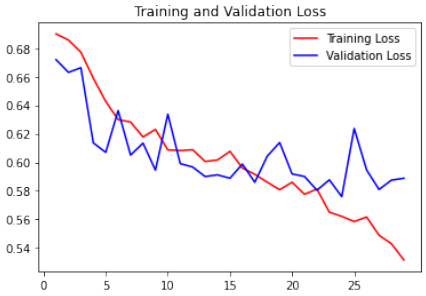

Figure 24: Model 2_2_de Loss on Training and Validation Set

As shown, the model doesn't overfit anymore. Training and validation accuracies present a slow but constant improvement, meaning that the model is learning. Results on the test set are the best ones reached so far: accuracy is 67% (approximately the same as before) while loss is 0.60 (it was 0.87).

Data Augmentation Another experiment that can be done is trying to augment the data. The strategy adopted is the one described in [She+19], in which only rotation (with $range = 90$) is performed. However the experiment produced a little decrease in the performances on the test set (67% vs. 64% on accuracy, 0.60 vs. 0.61 in loss). The main reasons of that are probably the same ones described in 2.1.2.

2.2.3 Experiment 2_3: Hyperparameters tuning and other optimizations

The aim of this experiment is to find possible optimizations of the model by varying the value of the hyperparameters and using particular weights initial-

ization strategies. The exploited approach is exactly the same already described in 2.1.3.

Conclusions More models obtained similar results to ones of the model developed in 2.2.2. Sometimes accuracy is lower and sometimes loss function is better, but these conditions are never true at the same time. So the model in 2.2.2 can still be considered the better obtained for this problem.

2.2.4 Conclusions

The best model is the one developed in 2.2.2, using Callback and Dropout. It reached an accuracy of 67% and the loss was 0.60.

3 Task 3: Usage of a Pre-Trained Network

The aim of this section is to build a model that exploits a pre-trained architecture (or architectures) to distinguish mammograms in Calcification or Mass cases (3.1) and in Benign or Malignant cases (3.2).

3.1 Subtask 1: Calcification-Mass Classification

This subsection is related to Calcification-Mass classification and it is organized as follows. Two experiments are done: feature extraction from *VGG16* and *ResNet50* (*v2* -Keras Website - is considered since it is easier to optimize, as described in [He+16]) and extraction of first layers only.
In order to do so, labels need to be redefined. At the current state, Calcification cases are characterized by labels 3 and 4, while Mass cases are characterized by labels 1 and 2. Label 0 is assigned to Calcification class and label 1 is assigned to Mass class. Class are balanced as shown in 2.1.

3.1.1 Experiment 1_1: Feature Extraction from VGG16 and ResNet50

The aim of this experiment is to evaluate the results of performing feature extraction from two of the most used ([TCP19], [XSG18], [She+19]) pre-trained networks, *VGG16* and *ResNet50*. Both networks are considered without their top layers ($include\_top = False$) and their weights are initialized according to the results of pre-training on *ImageNet*. Both bases are *frozen*, so the only layers that will be trained are the ones put on the top of those bases. As in 2.1.1 and 2.2.1, *validation_split* is set to 0.15, according to the value used in [XSG18] and [She+19].

Preprocessing Training and test images are normalized by dividing them for $MAX\_UINT16 = 65535$. Training images (and their respective labels) are shuffled because the parameter *validation_split* of $model.fit$ function will take the last portion of the dataset as validation set. To the current state, images (and their labels) are ordered by class (the elements labelled as 0 are at the end of the set). *Permutation* is used to apply the same reordering to both training images and labels. It is also important to check to composition of the validation set: 54.86% of the samples are Calcification ones, and 45.14% are Mass ones, so the two classes are equally balanced. No *Data Augmentation* is used in this experiment. Since pre-trained networks use three channels, the gray-scale value is replicated three times.

VGG16 *VGG16* is used as base in this experiment. A single randomly initialized fully-connected layer is put on the top of it (using two fully connected layers, as in [TCP19], led to no improvement), and its number of units is equal to 1024. The activation function is *ReLu*. Finally, a *Sigmoid* layer is used for classification. *Adam* is used as optimizer, and Learning Rate is set to 1e-5 (instead of 1e-6 of [TCP19], so the convergence will be faster). Because of the

low Learning Rate, the maximum number of epochs is increased to 1000, and a *Callback* with $patience = 15$ is used (see again [TCP19]).

Accuracy and loss on training and validation sets are shown in Figures 25-26.

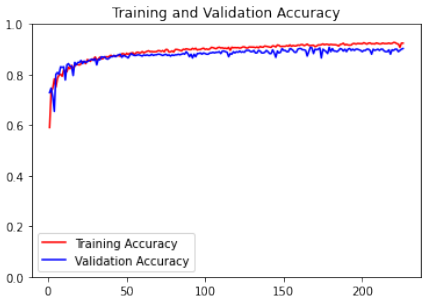

Figure 25: Model 1_1_vgg Accuracy on Training and Validation Set

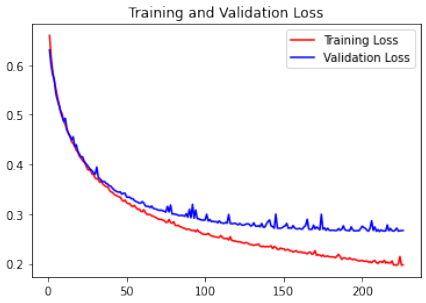

Figure 26: Model 1_1_vgg Loss on Training and Validation Set

Since the Learning Rate is low, the learning phase is very slow. This is highlighted by the fact that training and validation accuracy curve shows a small but continuous improvement, as well as training and validation loss curve. There is a very mild overfitting, as shown by the small difference between loss functions. However, results on the test set are in line with the best ones found in 2.1 ($accuracy = 85\%$, $loss = 0.34$). It can be also useful to evaluate some other metrics. For example, the confusion matrix of the test set samples is shown in Table 10.

		True Class	
		Calcification	Mass
Predicted Class	Calcification	131	24
	Mass	26	155

Table 10: Confusion Matrix for Test Set Samples evaluated on VGG16-based model (Feature Extraction)

As shown, the number of Calcification samples classified as Mass is almost equal to the number of Mass samples classified as Calcification. In addition, they represent only a small portion of the real Calcification and Mass samples (recalls are 0.83 and 0.87).
Finally, another indicator is the *ROC* Area Under the Curve (AUC) score: the value is 0.85, which is a satisfying result.

ResNet50 *ResNet50* is used as base in this experiment. In order to have a reasonable number of weights, a Global Average Pooling layer is used instead of a Flatten one. A single Fully Connected layer is used (the usage of more layers did not produce any improvement), and the number of hidden units is still fixed

to 1024. *ReLu* is used as Activation Function. Finally, a *Sigmoid* layer is used for classification. *Adam* is used as optimizer, and Learning Rate is set to 1e-5. Because of the low Learning Rate, the maximum number of epochs is increased to 1000, and a *Callback* with *patience* = 15 is used (see [TCP19]).

Accuracy and loss on training and validation sets are shown in Figures 27-28.

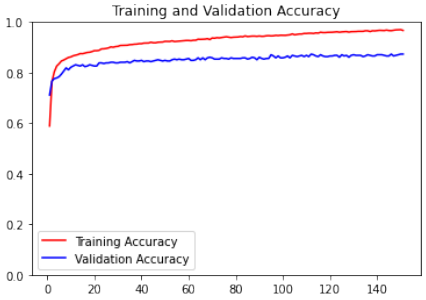

Figure 27: Model 1_1_res Accuracy on Training and Validation Set

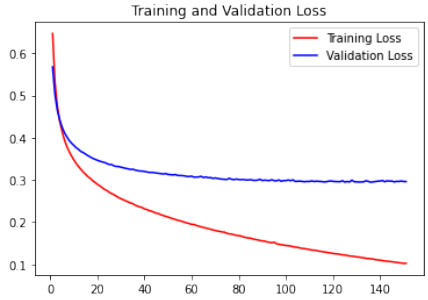

Figure 28: Model 1_1_res Loss on Training and Validation Set

The accuracy on the test set is 84%, while the loss reaches 0.40. These results are satisfying, however there is a little difference between training and validation accuracy and between training and validation loss and this can be an hint of possible overfitting. However, it is important to notice that validation accuracy is slowly increasing (and validation loss is slowly decreasing), meaning that the model is still learning.
Confusion matrix of the test set samples is shown in Table 11.

		True Class	
		Calcification	Mass
Predicted Class	Calcification	137	32
	Mass	20	147

Table 11: Confusion Matrix for Test Set Samples evaluated on ResNet50-based model (Feature Extraction)

With respect to *VGG16*-based results (Table 10), the distribution of the error is slightly unbalanced towards true Mass cases (the number of Mass cases classified as Calcification is greater than the number of Calcification cases classified as Mass). As a result, recalls assume opposite value with respect to the previous experiment (0.87 and 0.82).
AUC score (0.847) is very similar to the value obtained in *VGG16*-based experiment.

Conclusions *VGG16*-based model is slightly better than *ResNet50*-based, so the next experiment will consider only *VGG* as network base.

3.1.2 Experiment 1_2: Extract Only First Layers from VGG16

Since *ImageNet* dataset is far different from *CBIS-DDSM* dataset, an idea could be to use only the first pretrained layers of *VGG16*. As a matter of fact, they extract only the most generic features that can be reused in the current problem. The extracted layers are *frozen*, so that learned features are not modified. As in 3.1.1, *validation_split* is set to 0.15, according to the value in [XSG18] and [She+19].

Preprocessing Training and test images are normalized by dividing them for $MAX\_UINT16 = 65535$. Training images (and their respective labels) are shuffled because the parameter *validation_split* of *model.fit* function will take the last portion of the dataset as validation set. To the current state, images(and their labels) are ordered by class (the elements labelled as 0 are at the end of the set). *Permutation* is used to apply the same reordering to both training images and labels. It is also important to check to composition of the validation set: 54.86% of the samples are Calcification ones, and 45.14% are Mass ones,so the two classes are equally balanced. No Data Augmentation is used in this experiment. Since pre-trained networks use three channels, the gray-scale value is replicated three times.

VGG16 *VGG16* is used as base in this experiment, and only the first 4 blocks (each block is made up by two Convolutional layers and a Max Pooling layer) are extracted (the extraction of a lower number of layers decreased the performances). Then randomly initialized Convolutional layer and Max Pooling layer are added on the top of *VGG16* base. Finally, a *ReLu* Dense layer of 1024 units and a *Sigmoid* one are used to perform the final classification. *He Uniform* is used as *kernel_initializer* in *ReLu*-based layers (as described in TDS Website). *Adam* is used as optimizer, and Learning Rate is set to 1e-6 (as in [TCP19]). Because of the low Learning Rate, the maximum number of epochs is increased to 1000, and a *Callback* with *patience* = 15 is used (see again [TCP19]).
Accuracy and loss on training and validation sets are shown in Figures 29-30.

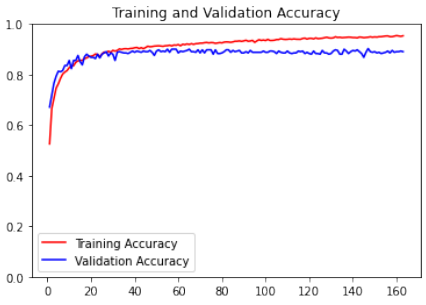

Figure 29: Model 1_2_vgg Accuracy on Training and Validation Set

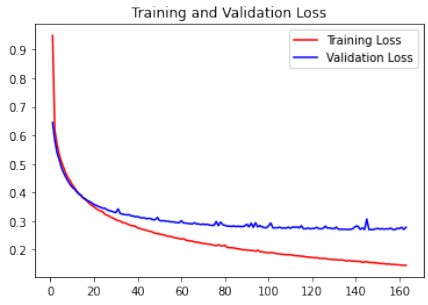

Figure 30: Model 1_2_vgg Loss on Training and Validation Set

Result on the test overcome the ones of the previous experiment. Accuracy is now 88% (it was 85%) and loss is 0.32 (it was 0.34). In addition, training phase is faster (160 epochs vs. 200). Training and validation accuracy approximately follow the same path, as well as training and validation loss. However, the improvement for validation curves seems slower than the training ones.
Confusion matrix is shown in Table 12.

		True Class	
		Calcification	Mass
Predicted Class	Calcification	133	16
	Mass	24	163

Table 12: Confusion Matrix for Test Set Samples evaluated on VGG16-based model (only first layers extracted)

With respect to the previous experiment (10, the model seems to perform better in the classification of Mass cases. As a matter of fact, recall is equal to 0.91. Error distribution is slightly unbalanced towards Calcification cases.
AUC score is better (0.88) is better than the ones previously obtained.

3.1.3 Conclusions

The experiment led to a significant improvement with respect to the results of the previous one, both in terms of results and performances.

3.2 Subtask 2: Benign-Malignant Classification

This subsection is related to Benign-Malignant classification and it is organized as follows. Two experiments are done: feature extraction from *VGG16* and *ResNet50* (*v2* -Keras Website - is considered since it is easier to optimize, as described in [He+16]) and extraction of first layers only.
In order to do so, labels need to be redefined. At the current state, Calcification cases are characterized by labels 3 and 4, while Mass cases are characterized by labels 1 and 2. Label 0 is assigned to Calcification class and label 1 is assigned to Mass class. Class are balanced as shown in 2.2.

3.2.1 Experiment 2_1: Feature Extraction from VGG16 and ResNet50

The aim of this experiment is to evaluate the results of performing feature extraction from two of the most used ([TCP19], [XSG18], [She+19]) pre-trained networks, *VGG16* and *ResNet50*. Both networks are considered without their top layers ($include\_top = False$) and their weights are initialized according to the results of pre-training on *ImageNet*. Both bases are *frozen*, so the only layers that will be trained are the ones put on the top of those bases. As in 2.1.1 and 2.2.1, *validation_split* is set to 0.15, according to the value used in [XSG18] and [She+19].

Preprocessing Training and test images are normalized by dividing them for $MAX\_UINT16 = 65535$. Training images (and their respective labels) are shuffled even if classes are already randomly distributed.*Permutation* is used to apply the same reordering to both training images and labels. It is also important to check to composition of the validation set: 62.84% of the samples are Benign ones, and 37.16% are Mass ones. No Data Augmentation is used in this experiment. Since pre-trained networks use three channels, the gray-scale value is replicated three times.

VGG16 *VGG16* is used as base in this experiment. A single randomly initialized fully-connected layer is put on the top of it (using two fully connected layers, as in [TCP19], led to no improvement), and its number of units is equal to 1024. The activation function is *ReLu*. Finally, a *Sigmoid* layer is used for classification. *Adam* is used as optimizer, and Learning Rate is set to 1e-5 (instead of 1e-6 of [TCP19], so the convergence will be faster with respect to a Learning rate of 1e-6). Because of the low Learning Rate, the maximum number of epochs is increased to 1000, and a *Callback* with *patience* = 7 is used (see again [TCP19]).

Accuracy and loss on training and validation sets are shown in Figures 31-32.

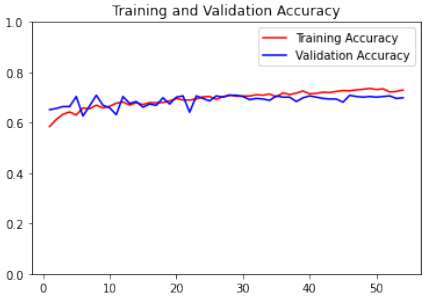

Figure 31: Model 2_1_vgg Accuracy on Training and Validation Set

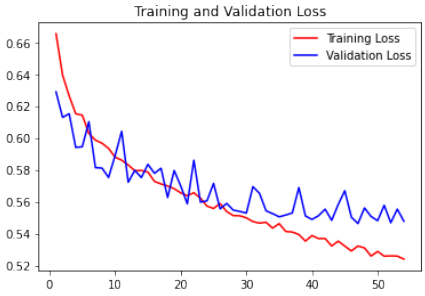

Figure 32: Model 2_1_vgg Loss on Training and Validation Set

Training phase is slower than from-scratch experiments. Improvement in validation accuracy is small but constant, and so it is for validation loss. However, results on the test set are slightly better than the ones in 2.2: accuracy is 68% (it was 67%) and loss is 0.59 (it was 0.60). It can be useful to evaluate other metrics, such as the confusion matrix (Table 13).

		True Class	
		Benign	Malignant
Predicted Class	Benign	181	68
	Malignant	38	49

Table 13: Confusion Matrix for Test Set Samples evaluated on VGG16-based model (Feature Extraction)

As shown, the distribution of the errors is unbalanced. The model has difficulty in classifying Malignant cases, as highlighted by the fact that recall (for Malignant cases) is only 0.42. The opposite error distribution is preferable, because classifying a Benign case as Malignant is better than classifying a Malignant case as Benign.
The AUC value is 0.62, which is in line with the results in [TCP19].

ResNet50 *ResNet50* is used as base in this experiment. In order to have a reasonable number of weights, a Global Average Pooling layer is used instead of a Flatten one. The number of hidden units is still fixed to 1024, and *ReLu* is used as Activation Function. Finally, a *Sigmoid* layer is used for classification. *Adam* is used as optimizer, and Learning Rate is set to 1e-6 (as described in [TCP19]). Because of the low Learning Rate, the maximum number of epochs is increased to 1000, and a *Callback* with *patience* = 15 is used (see [TCP19]).

Accuracy and loss on training and validation sets are shown in Figures 33-34.

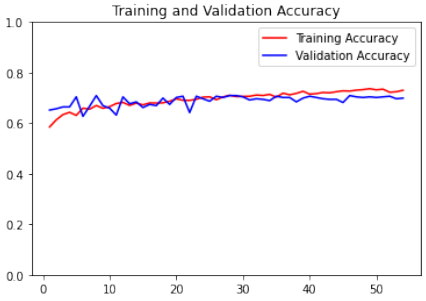

Figure 33: Model 2_1_res Accuracy on Training and Validation Set

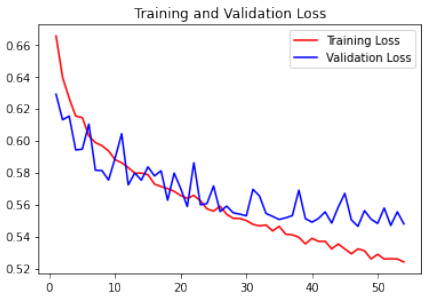

Figure 34: Model 2_1_res Loss on Training and Validation Set

Results on the test set are the best ones obtained so far (accuracy is 70%, loss is 0.56). Starting validation and training accuracy are already very high if compared with the final ones, so the improvement is slow but constant. Validation loss is decreasing in a slightly unstable way (however, values varies in a range lower than 0.02). It can be useful to evaluate other metrics, such as the confusion matrix (Table 14).

		True Class	
		Benign	Malignant
Predicted Class	Benign	166	48
	Malignant	53	69

Table 14: Confusion Matrix for Test Set Samples evaluated on ResNet50-based model (Feature Extraction)

With respect to *VGG16*-based model, this model is better at classifying Malignant cases. As a matter of fact, recall (Malignant) is now 0.59. On the other hand, the number of Benign cases correctly classified is slightly lower (166 vs. 181). AUC is now 0.67, perfectly in line with [TCP19].

Conclusions *ResNet50*-based model is better than the *VGG16*-based model, however due to its non-sequential nature it is difficult to perform first layers extraction only. Because of that, *VGG16* is considered for the next experiment.

3.2.2 Experiment 2_2: Extract Only First Layers from VGG16

Since *ImageNet* dataset is far different from *CBIS-DDSM* dataset, an idea could be to use only the first pretrained layers of *VGG16*. As a matter of fact, they extract only the most generic features that can be reused in the current problem. The extracted layers are *frozen*, so that learned features are not modified. As in 3.1.1, *validation_split* is set to 0.15, according to the value in [XSG18] and [She+19].

Preprocessing Training and test images are normalized by dividing them for $MAX\_UINT16 = 65535$. Training images (and their respective labels) are shuffled even if classes are already randomly distributed.*Permutation* is used to apply the same reordering to both training images and labels. It is also important to check to composition of the validation set: 62.84% of the samples are Benign ones, and 37.16% are Mass ones. No Data Augmentation is used in this experiment. Since pre-trained networks use three channels, the gray-scale value is replicated three times.

VGG16 *VGG16* is used as base in this experiment, and only the first 4 blocks (each block is made up by two Convolutional layers and a Max Pooling layer) are extracted (the extraction of a lower number of layers led to a decrease in the performances). Then randomly initialized Convolutional layer and Max Pooling layer are added on the top of *VGG16* base. A Dropout layer (with $rate = 0.25$) is used to avoid overfitting. Finally, a *ReLu* Dense layer of 1024 units and a *Sigmoid* one are used to perform the final classification. *He Uniform* is used as *kernel_initializer* in *ReLu*-based layers (as described in TDS Website). *Adam* is used as optimizer, and Learning Rate is set to 1e-6 (as in [TCP19]). Because of the low Learning Rate, the maximum number of epochs is increased to 1000,

and a *Callback* with *patience* = 10 is used (see again [TCP19]).
Accuracy and loss on training and validation sets are shown in Figures 35-36.

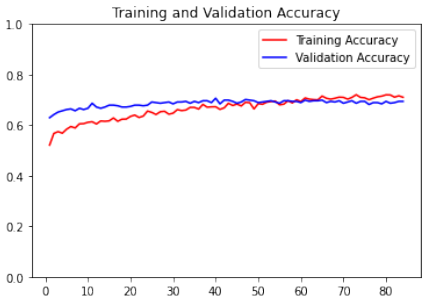

Figure 35: Model 2_2_vgg Accuracy on Training and Validation Set

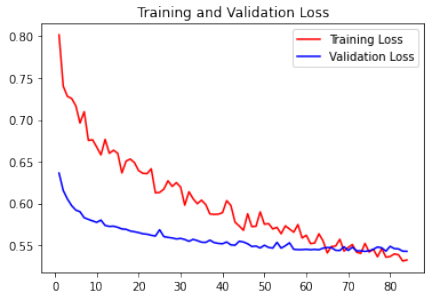

Figure 36: Model 2_2_vgg Loss on Training and Validation Set

Results on the test set are the best obtained so far: accuracy is 70% while loss is 0.57. It is important to notice that validation loss is lower than training loss for the greater part of the training phase: as described in Keras Website, Dropout (and other Regularization techniques) are turned off at testing time. So validation loss (and the same for accuracy) is more stable and can achieve better results.
Confusion matrix is shown in Table 15.

		True Class	
		Benign	Malignant
Predicted Class	Benign	179	60
	Malignant	40	57

Table 15: Confusion Matrix for Test Set Samples evaluated on ResNet50-based model (Feature Extraction)

With respect to the previous experiment, the model is weak in identifying Malignant cases (recall is only 0.49), but on the other hand is stronger in identifying Benign cases (recall is 0.82). AUC score is slightly lower than the value obtained in *ResNet50* experiment (0.65 vs. 0.67).

3.2.3 Conclusions

Results are very similar to the ones obtained from *ResNet50* based model in the previous experiment. In particular, the models seem to be complementary, so they can be used for further experiments.

4 Task 4: Usage of Baseline Patches

The aim of this section is to exploit Baseline patches in order to build a classifier that distinguish between Calcification and Mass or between Benign and Malignant cases. This section is focused on the latter classification problem.

4.1 Subtask 1: Benign-Malignant Classification

As mentioned above, this subsection is focused on Benign-Malignant classification problem only. The problem reached worse results if compared to Calcification-Mass problem, and because of that it would be the one with the greatest room for improvement.
In order to do so, labels need to be redefined. At the current state, Benign cases are characterized by labels 1 and 3, Malignant cases are characterized by labels 2 and 4 and finally Baselines are characterized by the label 0. Label 0 is assigned to Benign cases and label 1 is assigned to Malignant cases, as in previous experiments.

4.1.1 Experiment 1: Feeding Models with Baselines for Benign-Malignant Classification - from-Scratch models

In order to have a greater number of images to be used in the training phase, baseline patches can be exploited. In this experiment, each training baseline is considered as part of the class of the corresponding abnormality patch (namely Benign or Malignant). The idea is that the features extracted from the baselines can be useful to perform the classification as well as the ones extracted from abnormality patches. Baselines are considered in training set only, while validation and test set are free of them. This is because in real classification phase only abnormality patches are considered, and baselines are not.
In this experiment, only from-scratch models are considered.

Preprocessing Baselines and abnormality patches for training are splitted into validation and training set using 0.2 as validation split. In validation set, baselines are then discarded leading to a validation-training ratio of approximately 12%. Test baselines are not considered for this experiment. To sum up, training set is made up by 2468 (57.64%) Benign and 1814 (42.36%) Malignant cases (baselines and abnormality patches). Validation set is made up by 334 (62.43%) Benign cases and 201 (37.57%) Malignant cases (abnormality patches only). Finally, test set is made up by 219 (65.18%) Benign cases and 117 (34.82%) Malignant cases (abnormality patches only).
Training, validation test images are normalized by dividing them for $MAX\_UINT16 = 65535$. The gray-scale value is replicated three times in order to have three channels.

Model 4_a This model is inspired by *AlexNet*, described in [KSH12]. It is made up by two convolutional blocks (a Convolutional layer and a Max Pool-

ing layer), followed by three additional Convolutional layers and a Max Pooling layer. A Dropout layer ($rate = 0.25$) is used to avoid overfitting.
Sigmoid is used as activation function in the last Fully Connected layer, while *ReLu* is used in all the others Convolutional and Fully Connected layers. $Padding = same$ ensure that Convolutional layers do not perform size reduction. The overall structure is summed up in Figure 37.

Layer (type)	Output Shape	Param #
CONV_1 (Conv2D)	(None, 48, 48, 96)	23424
MAX_POOL_1 (MaxPooling2D)	(None, 24, 24, 96)	0
CONV_2 (Conv2D)	(None, 24, 24, 256)	614656
MAX_POOL_2 (MaxPooling2D)	(None, 12, 12, 256)	0
CONV_3 (Conv2D)	(None, 12, 12, 512)	1180160
CONV_4 (Conv2D)	(None, 12, 12, 512)	2359808
CONV_5 (Conv2D)	(None, 12, 12, 256)	1179904
MAX_POOL_3 (MaxPooling2D)	(None, 5, 5, 256)	0
FLAT_1 (Flatten)	(None, 6400)	0
DROP_1 (Dropout)	(None, 6400)	0
DENSE_1 (Dense)	(None, 1024)	6554624
DENSE_2 (Dense)	(None, 1024)	1049600
DENSE_3 (Dense)	(None, 1)	1025

Total params: 12,963,201
Trainable params: 12,963,201
Non-trainable params: 0

Figure 37: Model 4_a summary

In addition to the Dropout layer, a *Callback* with *patience* = 15 is used to perform the early stopping of the training. *Adam* is used as optimizer, and Learning Rate is set to 1e-6. Since the total number of training images is greater than the previous experiments, an higher *batch size* is used (64). Finally, the maximum number of epochs is set to 1000.
Training and validation accuracy and loss are shown in Figures 38 and 39.

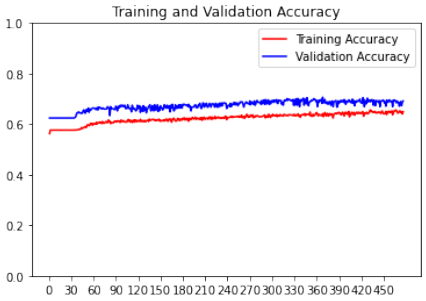

Figure 38: Model 4_a Accuracy on Training and Validation Set

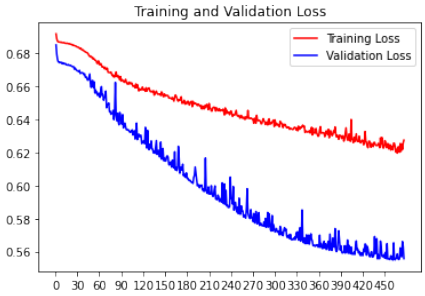

Figure 39: Model 4_a Loss on Training and Validation Set

As shown, accuracy improvement is slow but continuous. The fact that validation accuracy is better than training accuracy is due to the regularization technique used (Dropout), that is applied in training but not in validation phase. The model is therefore more robust in validation phase and because of that it obtains better results in that phase. For the same reason, validation loss is lower than training loss. Another possible reason can be that the validation set is simpler, because it contains only abnormality patches and not baselines
Results on the test set are in line with the ones achieved by previous experiments (accuracy is 68% while loss is 0.60). However, if compared to the result achieved by *Model1_1_a* (2.2.1), which exploits a similar architecture, the improvement is consistent (accuracy from 60% to 68%), even if the improvement itself is not fully imputable to the usage of baselines.
Confusion matrix is shown in Table 16.

		True Class	
		Benign	Malignant
Predicted Class	Benign	174	62
	Malignant	45	55

Table 16: Confusion Matrix for Test Set Samples evaluated on Model 4_a

As shown, the model is good in identifying Benign cases but it has some difficulties in correctly evaluating Malignant cases. As a matter of fact, recall (Malignant) is only 0.47. AUC score is equal to 0.6323, which is in line with previous experiment and the values achieved in [TCP19].

Model 4_b This model is obtained by simply stacking four Convolutional blocks, each one made up by a Convolutional layer and a Max Pooling layer. A Dropout layer (with $rate = 0.25$) is used to avoid overfitting. Finally, a single Fully Connected layer is added together with the layer used for the classification. Except for the latter, that is *Sigmoid* based, all the other layers (Convolutional

and Fully Connected) are *ReLu*-based. The overall structure is summed up in Figure 40.

Layer (type)	Output Shape	Param #
CONV_1 (Conv2D)	(None, 148, 148, 64)	1792
MAX_POOL_1 (MaxPooling2D)	(None, 74, 74, 64)	0
CONV_2 (Conv2D)	(None, 73, 73, 96)	24672
MAX_POOL_2 (MaxPooling2D)	(None, 36, 36, 96)	0
CONV_3 (Conv2D)	(None, 34, 34, 192)	166080
MAX_POOL_3 (MaxPooling2D)	(None, 17, 17, 192)	0
CONV_4 (Conv2D)	(None, 15, 15, 192)	331968
MAX_POOL_4 (MaxPooling2D)	(None, 5, 5, 192)	0
FLAT_1 (Flatten)	(None, 4800)	0
DROP_1 (Dropout)	(None, 4800)	0
DENSE_1 (Dense)	(None, 1024)	4916224
DENSE_2 (Dense)	(None, 1)	1025

Total params: 5,441,761
Trainable params: 5,441,761
Non-trainable params: 0

Figure 40: Model 4_b summary

For the training phase, the same hyperparameters of 4.1.1 are used: *Adam* as optimizer, 1e-6 as Learning Rate, 64 as *batch size*, 1000 as Maximum number of Epochs. Again, a *Callback* with *patience* = 15 is used.
Training and validation accuracy and loss are shown in Figures 41 and 42.

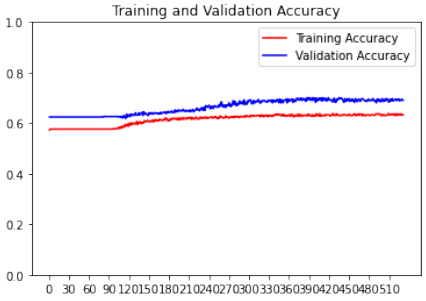

Figure 41: Model 4_b Accuracy on Training and Validation Set

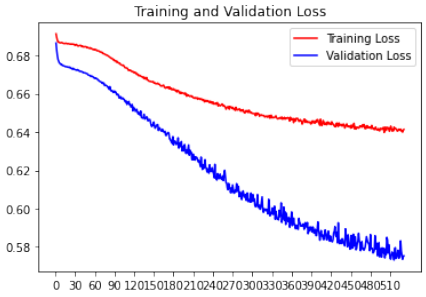

Figure 42: Model 4_b Loss on Training and Validation Set

Again, validation accuracy and loss are better than training ones. This is probably for the same reason explained in 4.1.1. However, results are slightly

worse than the ones from the previous experiment: accuracy is 67% and loss is 0.61.
Confusion matrix is shown in Table 17.

		True Class	
		Benign	Malignant
Predicted Class	Benign	179	70
	Malignant	40	47

Table 17: Confusion Matrix for Test Set Samples evaluated on Model 4_b

The same considerations done for 4.1.1 are valid for this model. Again, Malignant recall is even worse than the one in 4.1.1 (0.40). The fact the model is worse than thee preceding one is also highlighted by the lower AUC score (0.61).

Conclusions The two models obtained similar results. In the first case, the results were better than the similar base model (2.1.1, but not completely imputable to baselines and anyway in line with results from other experiments), while in the second case they are similar. Both the models presents difficulties in correctly classifying Malignant cases. To sum up, the usage of baselines did not produce consistent improvement in the classification between Benign and Malignant cases.

4.1.2 Experiment 2: Feeding Models with Baselines for Benign-Malignant Classification - pre-trained models

For this experiment, pre-trained models are considered. The idea is still that the usage of baselines in training phase can have some benefits, for example the fact that the training set can have more images available. Pre-trained networks are imported without their top layers ($include\_top = false$) and the base is *frozen*, so the only trainable layers are the ones put on the top of the base.

Preprocessing Preprocessing used for this experiment is the same used for the previous experiment, and it is described in 4.1.1.

VGG16 *VGG16* is used as base in this experiment. A Fully Connected layer is put on the top of the base (*ReLu* is used as activation function), and finally the classification layer (*Sigmoid*) is added. In order to avoid overfitting, a Dropout layer ($rate = 0.25$) and a Callback ($patience = 15$) are used. *Batch size* is again set to 64, maximum number of Epochs is set to 1000 and *Adam* is used as optimizer ($learning\_rate = 1e-6$).
Training and validation accuracy and loss are shown in Figures 43 and 44.

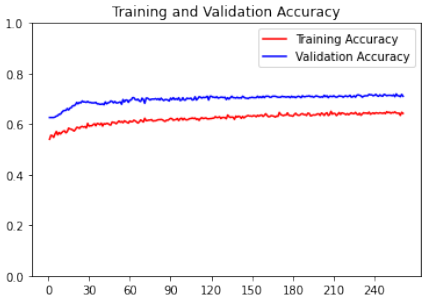

Figure 43: Model 4_vgg Accuracy on Training and Validation Set

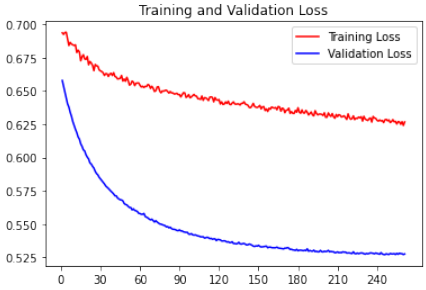

Figure 44: Model 4_vgg Loss on Training and Validation Set

As in 4.1.1 and 4.1.1, validation results are better than training ones. However in this case the difference is more visible and consistent, probably because of the combined effect of regularization and validation set simplicity.
Test accuracy is 68% while loss is 0.61, which are common results in the experiments previously done.
Confusion matrix is shown in Table 18.

		True Class	
		Benign	Malignant
Predicted Class	Benign	176	65
	Malignant	43	52

Table 18: Confusion Matrix for Test Set Samples evaluated on Model 4_vgg

Again, there is a huge difference in Benign recall (0.80) and Malignant recall (0.44), and this highlights the difficulty of the model in correctly evaluating Malignant cases. AUC score is in line with the values from previous experiments and the results in [TCP19].

ResNet50 *ResNet50* is used as base in this experiment. In particular, *V2* is considered ([He+16]) since it is easier to optimize. In order to reduce the number of trainable weights, a Global Average Pooling is used instead of a Flatten one. Then, a Fully Connected layer is added (*ReLu* based) and finally another one to perform the classification (*Sigmoid*). In order to avoid overfitting, a Dropout layer ($rate = 0.25$) and a Callback ($patience = 15$) are used. *Batch size* is again set to 64, maximum number of Epochs is set to 1000 and *Adam* is used as optimizer ($learning\_rate = 1e-6$).
Training and validation accuracy and loss are shown in Figures 45 and 46.

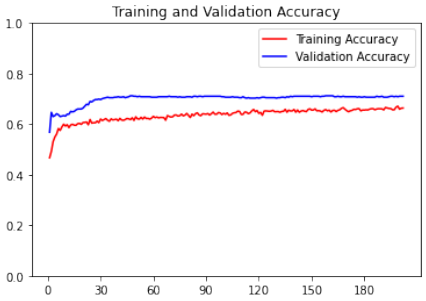

Figure 45: Model 4_res Accuracy on Training and Validation Set

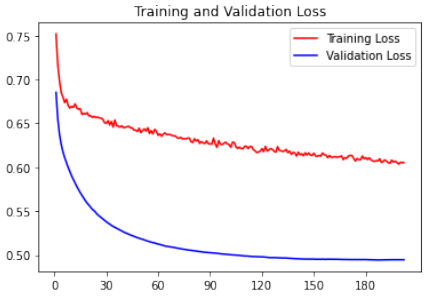

Figure 46: Model 4_res Loss on Training and Validation Set

The same considerations done in 4.1.2 can be done for this model, in terms of training and validation loss and accuracy. The main difference between the models is in terms of training time: the total number of epochs needed for the training is 202 (vs. 260 in 4.1.2) and the time spent for each epoch is 7s (vs. 9 in 4.1.2). Accuracy on the test set is 67% and loss is 0.58, so even this model does not present consistent differences with the previous ones.
Confusion matrix is shown in Table 19.

		True Class	
		Benign	Malignant
Predicted Class	Benign	172	63
	Malignant	47	54

Table 19: Confusion Matrix for Test Set Samples evaluated on Model 4_res

The error is again slightly unbalanced towards Malignant cases (recall only a few decimals better than 4.1.2, 0.46 vs. 0.44). AUC score is in line with previous results (0.62).

Conclusions Even in the case of pre-trained networks, the usage of baselines did not produce significant improvement. Accuracy and loss values are in line with the ones of the experiments without baselines.

4.1.3 Conclusions

The usage of baselines in training phase did not produce any improvement on the Benign-Malignant classification task. On the other hand, it did not produce any decrease in the performances, as one might expect by feeding the model with non-abnormality patches. Apparently, baselines do not contain features useful for the Benign-Malignant classification problem.

5 Task 5: Developement of a Composite Classifier

The aim of this section is to build a composite classifier that exploits already developed models in order to boost classification performance. Different approaches can be adopted: majority voting, average voting and weighted average voting.

5.1 Subtask 1: Calcification-Mass Classification

This subsection is focused on Calcification-Mass classification. In order to perform a significant and effective experiment, the three best models from previous experiments are considered. They are summed up in Table 20.

Models Considered for Calcification-Mass Ensemble		
Model	Test Set Accuracy	Test Set Loss
calcification_mass_scratch	87.5%	0.34
calcification_mass_pretrained_vgg1	85%	0.35
calcification_mass_pretrained_vgg2	88%	0.32

Table 20: Models Considered for Calcification-Mass Ensemble

As in previous experiments, label 0 is assigned to Calcification while label 1 is assigned to Mass cases.

5.1.1 Experiment 1_1: Majority Voting

In majority voting, each classifier makes its own class prediction and the class which achieves the greatest number of votes will be the one predicted by the composite classifier.
In this specific case, since three classifier are considered, it is sufficient to sum the predictions (0 or 1) of all classifier and assign label 1 if the sum is equal to 2 or 3 and label 0 otherwise.
Accuracy on the test set is equal to 86%: that value is in between the values of the base classifiers. In conclusion, Majority Voting does not improve the accuracy of the best classifier (88%) because, for example, wrong predictions of that classifier are not corrected by the predictions of the other two.
Confusion Matrix is shown in 21.

		True Class	
		Calcification	Mass
Predicted Class	Calcification	130	19
	Mass	27	160

Table 21: Confusion Matrix for Majority Voting in Calcification-Mass problem

Recall are both very high (0.83 and 0.89), meaning that the model is still good for this classification problem. With respect to *calcification_mass_pretrained_vgg2* (3.1.2), this model loses the correct classification label for 3 Benign cases and 3 Malignant cases. This is probably because of the reason explained above: the other two models used assigned to that case the opposite label, and so does the composite classifier (2 votes against 1).

5.1.2 Experiment 1_2: Average Voting

In average voting, each classifier makes its own raw prediction and the average of all of them is then computed. The class is finally deducted from the average prediction.
Accuracy on the test set reaches 87%, which is again slightly lower with respect to *calcification_mass_pretrained_vgg2*. It can be useful to visualize the Confusion Matrix (Table 22).

		True Class	
		Calcification	Mass
Predicted Class	Calcification	130	16
	Mass	27	163

Table 22: Confusion Matrix for Average Voting in Calcification-Mass problem

As shown, Confusion Matrix is very similar to the one of Majority Voting (21). However in this case, since the average is more robust, the prediction of Mass cases is the same of the best model (3.1.2). This is probably because *calcification_mass_pretrained_vgg2* predictions (for the three misclassified Mass cases in Majority Voting) are close to 1 and this makes difficult to make the average lower than 0.5, even if the other two classifier assign to the cases a 0 label.

5.1.3 Experiment 1_3: Weighted Average Voting

This experiment is very similar to the former one, but weights are assigned to each classifier according to the results obtained on the test set. In particular, *accuracy* and *loss* are used to compute weights according to the following formula:

$$weight = \frac{accuracy * 100}{loss} \tag{1}$$

Weights are rounded up to the second decimal digit. In particular, obtained weights are the following:

- **calcification_mass_scratch**: 255.26
- **calcification_mass_pretrained_vgg1**: 245.17
- **calcification_mass_pretrained_vgg2**: 274.03

The averaged predictions from each classifier are summed and then divided by the sum of the weights.
Accuracy on test set reaches 87% in this experiment. Confusion Matrix is shown in the following Table (23).

		True Class	
		Calcification	Mass
Predicted Class	Calcification	130	16
	Mass	27	163

Table 23: Confusion Matrix for Weighted Average Voting in Calcification-Mass problem

As shown, Confusion Matrix is identical with respect to the one from the previous experiment (Average Voting). So the usage of different weights did not produce any improvement in terms of overall accuracy on the test set. A possible reason is that the weights are not so different, so the final result is similar to the one obtained considering equal weights for all classifiers. Another reason can be that errors are too close to the labels (0 or 1) to be corrected from a simple weighted average.

5.1.4 Conclusions

For Calcification-Mass problem, the developement of a composite classifier led to unsatisfactory results. All the test set accuracy obtained are slightly lower than the accuracy of the best classifier (88%). A possible reason can be that all the considered base classifiers made errors on the same subset of cases, and for that reason a composite classifier is not helpful. Another possibility is that errors are too close to 0 or 1 to be corrected by the average, for example.

5.2 Subtask 2: Benign-Malignant Classification

This subsection is focused on Benign-Malignant classification. In order to perform a significant and effective experiment, the three best models from previous experiments are considered. They are summed up in Table 24.

Models Considered for Benign-Malignant Ensemble		
Model	Test Set Accuracy	Test Set Loss
benign_malignant_scratch	68%	0.60
benign_malignant_pretrained_vgg2	70%	0.57
benign_malignant_pretrained_res1	70%	0.56

Table 24: Models Considered for Benign-Malignant Ensemble

As in previous experiments, label 0 is assigned to Benign while label 1 is assigned to Malignant cases.

5.2.1 Experiment 2_1: Majority Voting

In majority voting, each classifier makes its own class prediction and the class which achieves the greatest number of votes will be the one predicted by the composite classifier.
In this specific case, since three classifier are considered, it is sufficient to sum the predictions (0 or 1) of all classifier and assign label 1 if the sum is equal to 2 or 3 and label 0 otherwise.
Accuracy on the test set is equal to 71%, which is slightly better with respect to the accuracy of all the base classifier used to build to composite classifier. This means that the wrong predictions of a classifier are corrected by the accurate predictions of the other two. It can be useful to examine the Confusion Matrix (Table 25).

		True Class	
		Benign	Malignant
Predicted Class	Benign	188	68
	Malignant	31	49

Table 25: Confusion Matrix for Majority Voting in Benign-Malignant problem

As shown, the model is still in difficulty in correctly classifying Malignant cases (recall is only 0.42). With respect to *benign_malignant_pretrained_res1* (the best base model, 3.2.1), the main difference is that this model is far better in Benign classification but is defective in Malignant classification. As a matter of fact, Benign recalls are 0.76 (*benign_malignant_pretrained_res1*) and 0.86 (this model), while Malignant recalls are 0.59 (*benign_malignant_pretrained_res1*) and 0.42 (this model). This means that the other two models used in the composite classifier (*benign_malignant_pretrained_vgg2* and *benign_malignant_scratch*) disclaim the correct Malignant classifications from *benign_malignant_pretrained_res1*, but at the same time they are able to correct its wrong Benign classifications.

5.2.2 Experiment 2_2: Average Voting

In average voting, each classifier makes its own raw prediction and the average of all of them is then computed. The class is finally deducted from the average prediction.
Accuracy is 72%, which is better than the values from previous experiment and from base classifiers. Again, it can be useful to evaluate the Confusion Matrix (Table 26).

		True Class	
		Benign	Malignant
Predicted Class	Benign	187	62
	Malignant	32	55

Table 26: Confusion Matrix for Average Voting in Benign-Malignant problem

Benign and Malignant recalls are 0.85 and 0.47 respectively. Again, with respect to the best base model (*benign_malignant_pretrained_res1*, 3.2.1), the model is better in classifying Benign cases but is worse in classifying Malignant cases. However, the robustness of the average with respect to the majority voting led to a better Malignant classification compared to the Majority voting experiment (Malignant recalls are 0.42 for Majority voting and 0.47 for Average Voting).

5.2.3 Experiment 2_3: Weighted Average Voting

This experiment is very similar to the former one, but weights are assigned to each classifier according to the results obtained on the test set. In particular, *accuracy* and *loss* are used to compute weights according to the following formula:

$$weight = \frac{accuracy * 100}{loss} \tag{2}$$

Weights are rounded up to the second decimal digit. In particular, obtained weights are the following:

- **benign_malignant_scratch**: 112.26
- **benign_malignant_pretrained_vgg2**: 123.02
- **benign_malignant_pretrained_res1**: 124.08

The averaged predictions from each classifier are summed and then divided by the sum of the weights.
Accuracy on test set reaches 73% in this experiment. Confusion Matrix is shown in the following Table (27).

		True Class	
		Benign	Malignant
Predicted Class	Benign	186	59
	Malignant	33	58

Table 27: Confusion Matrix for Weighted Average Voting in Benign-Malignant problem

As shown, classification count for Benign cases is almost the same of the Average Voting experiment, but Malignant classification is improved up to a 50% recall. This means that the fact of assigning different weights to the classifier led to good results. In particular, a good tradeoff between the good performances of *benign_malignant_pretrained_res1* in Malignant classification and the good performances of *benign_malignant_scratch* and *benign_malignant_pretrained_vgg2* in Benign classification is reached because of the higher weight assigned to *benign_malignant_pretrained_res1*.

5.2.4 Conclusions

For Benign-Malignant classification problem, the developement of a composite classifier led to good results. All the experiments led to an improvement of the results with respect to the base classifiers. The best result is reached from the Weighted Average Voting, which permitted to pass from 70% to 73% in accuracy. In general, Average Voting classifiers provided better results than Majority Voting because of their higher robustness.

References

[TCP19] Lazaros Tsochatzidis, Lena Costaridou, and Ioannis Pratikakis. "Deep Learning for Breast Cancer Diagnosis from Mammograms—A Comparative Study". In: *Journal of Imaging* (2019). DOI: https://doi.org/10.3390/jimaging5030037.

[XSG18] Pengcheng Xi, Chang Shu, and Rafik Goubran. "Abnormality Detection in Mammography using Deep Convolutional Neural Networks". In: *IEEE International Workshop on Medical Measurement and Applications* (2018). DOI: https://doi.org/10.1109/MeMeA.2018.8438639.

[She+19] Li Shen et al. "Deep Learning to Improve Breast Cancer Early Detection on Screening Mammography". In: *Nature* (2019). DOI: https://doi.org/10.1038/s41598-019-48995-4.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Communications of the ACM* (2012). DOI: https://doi.org/10.1145/3065386.

[SZ15] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Neural Networks for Large-Scale Image Recognition". In: *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)* (2015). DOI: https://doi.org/10.1109/ACPR.2015.7486599.

[He+16] Kaiming He et al. "Identity Mappings in Deep Residual Networks". In: *Computer Vision – ECCV 2016* (2016). DOI: https://doi.org/10.1007/978-3-319-46493-0_38.