Media Engineering and Technology Faculty
German University in Cairo

Random Forest for Mammogram-based Cancer Detection

Bachelor Thesis

Author: Mayar Mohamed Alfares
Supervisors: Assoc. Prof. Mohammed Abdel-Megeed Salem
Submission Date: 26 May, 2019

Media Engineering and Technology Faculty
German University in Cairo

Random Forest for Mammogram-based Cancer Detection

Bachelor Thesis

Author: Mayar Mohamed Alfares
Supervisors: Assoc. Prof. Mohammed Abdel-Megeed Salem
Submission Date: 26 May, 2019

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowledgment has been made in the text to all other material used

---

Mayar Alfares
26 May, 2019

Acknowledgments

Throughout the writing of this thesis and the project implementation I have received a great deal of support and assistance.
I would like to thank Assoc. Prof. Mohammed Abdel-Megeed Salem for the continuous supervision of my bachelor project, for his patience, motivation and knowledge.
I would also like to express my sincere gratitude to my family and friends for showing me support along this journey.

Abstract

In this thesis, we addressed the problem of classifying mammograms into normal, benign and malignant.

Two cascaded classification tasks were implemented using different feature extractors, feature reducers and classifiers. The first classification phase (normal vs abnormal) achieved an accuracy of 95.27 percent and a cross-validation score of 97.85 while the second classification phase (benign vs malignant) resulted in a 70 percent accuracy and a cross-validation score of 83.32 despite the limitations found in the relatively small number of abnormal samples. Additionally, the runtime for a mammogram classification was decreased to 1.62 seconds on average.

The main focus of our thesis was to achieve state-of-the-art in detecting abnormalities while optimizing the runtime for a user-friendly interface.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Breast Cancer is the second most common cancer among females and it consists one of the leading cause of death worldwide, as stated by the World Cancer Research Fund. Mammography is the standard routine used for breast cancer screening. It provides high quality images that represent the internal anatomy of the breast. The potential of treatment is therefore increased due to mammogram-based early detection and accurate diagnosis.

Mammography screening has been proved to reduce breast cancer risk of mortality by 3848 percent among participants, according to Broeders, M. et al [3].

However, mammograms are vulnerable to the human error caused by radiologists wrong interpretations. Consequently, in order to improve the biopsies diagnostic efficiency, double-blind reading is performed to provide a second opinion and to reduce, to some extent, the workload on the radiologist. Nevertheless, this strategy consumes a lot of resources between money, time and labor, and it remains subject to errors.

Several computer-aided methodologies have been developed in mammogram detection (CADe) and diagnosis (CADx). The technology was approved by the FDA and is currently put into practice as an alternative for the second reader in the double-blind workflow to assist in and support the decisions made.

Gromet [23], in his study, showed that a second reader increases the sensitivity rates from 81.4 to 88.0 percent. Meanwhile, the sensitivity of a CAD reading scored 90.4 percent.

1.2 Problem Statement

In this thesis, we aim to present an approach to detect mammogram masses and classify them into normal, benign and malignant tumors using Random Forest (RF), one of the most powerful Machine Learning techniques that recently established state-of-the-art in the fields of image representation and object recognition.

1.3 Thesis Outline

The second chapter of this thesis contains the literature review along an overview of the main concepts.
The methodology is present in chapter three. It is implemented in five main steps: Image Pre-processing, Extraction of Regions of Interest (ROIs), Feature Extraction, Feature Reduction and Mammogram Classification as shown in figure 1.1 .

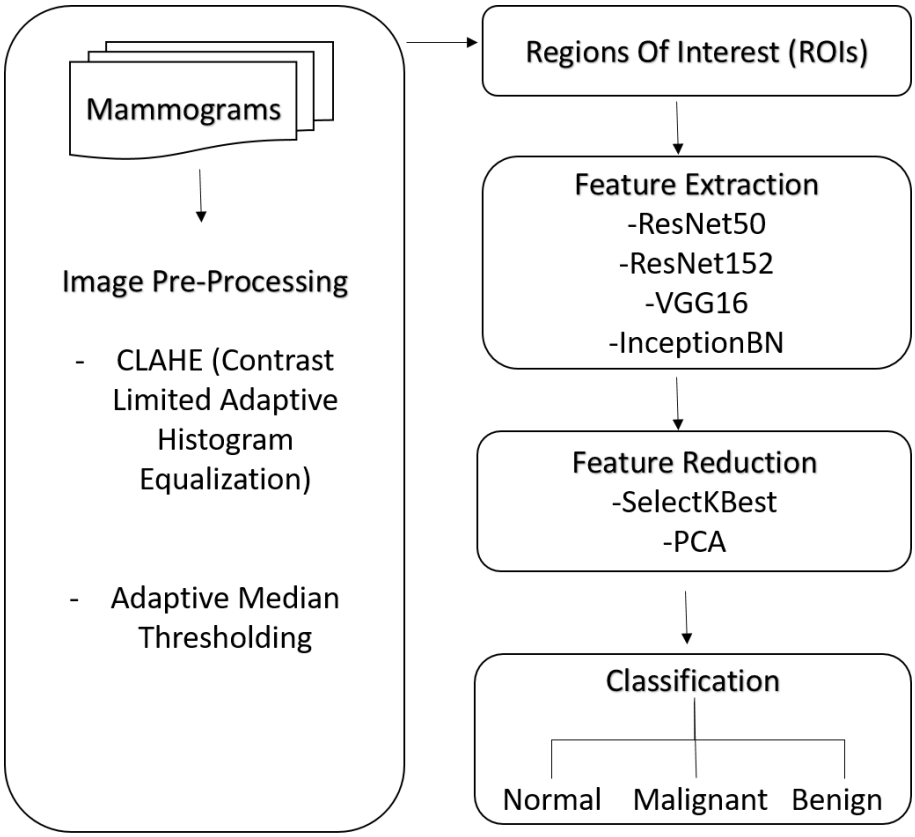


Figure 1.1: Workflow of mammograms classification

Finally, the experimental results are briefed in chapter four, while chapter five includes the thesis conclusion and future work.

Chapter 2

Background

2.1 Literature Review

2.1.1 Datasets

The mini-MIAS, DDSM, INbreast and BCDR datasets were included in the previous studies.

The mini-MIAS database [10]: The Mammographic Image Analysis Society is an organization located in the UK. The society collected 322 digitized images from the UK National Breast Screening Programme, 1024x1024 pixels each. Truth-markings for locations of present abnormalities are attached. In addition, the MIAS database offered some corresponding information of lesion area such as type, location, severity, central coordinate, and radius. Classes of abnormality present are calcification, well-defined/circumscribed masses, spiculated masses, other ill-defined masses, architectural distortion, asymmetry and normal. Severity of abnormality is classified as benign or malignant tumor. The dataset is widely covered in many papers due to its public and free availability via the Pilot European Image Processing Archive (PEIPA) at the University of Essex.

Digital Database for Screening Mammography (DDSM) [5]: DDSM represents an alternative resource for possible use by the research community interested in mammographic image analysis. It has been generated by a cooperation between Massachusetts General Hospital, Sandia National Laboratories and the University of South Florida Computer Science and Engineering Department. DDSM contains a set of 2,500 studies. Two images of each breast are included per study and each breast consists of a mediolateral oblique and craniocaudal view mammogram. Each study also comprehend the corresponding patient information (age, ACR breast density rating, subtlety rating for abnormalities, ACR keyword description of abnormalities) and image information (scanner, spatial resolution, ...). Pixel-level "ground truth" information are also included for images with suspicious regions. The cases have four pathology categories: normal, cancer, benign, and benign without callback.

INbreast database [28]: The INbreast database comprehends images acquired at a Breast Centre, situated in a University Hospital (Hospital de So Joo, Breast Centre, Porto, Portugal). The dataset covers 115 cases (410 images): 90 cases are taken of women with both breasts (4 images per case, 2 images CC and MLO per breast) in addition to 25 cases taken from mastectomy patients (only 2 images per case). Several types of lesions (masses, calcifications, asymmetries, and distortions) exist and are contoured by specialists. Unlike the previously mentioned datasets, the truth markings are not annotated as normal, benign and malignant. Mammograms are labeled in conformity with the Bi-Rads score. BI-RADS stands for Breast Imaging Reporting and Data System. It is a scheme for mammogram screening reports categorized into 7 well-defined divisions.

Breast Cancer Digital Repository (BCDR) [25]: The BCDR repository was created by the IMED Project (for Development of Algorithms for Medical Image Analysis) in order to establish a medical image repositories and to produce some Computer-Aided Diagnosis (CADx) methods related to GRID computing. Currently, the BCDR comprises 1734 cases of patients with mammography and ultrasound images, clinical history, lesion segmentation and selected pre-computed image-based descriptors. Mammograms are classified according to BIRADS and annotated by specialized radiologists. The respective ROIs are provided: 426 benign mass lesions and 310 malign mass lesions. The images contain both CC and MLO views for each breast (left and right). The dataset is publicly available to registered users.

Final Assessment Categories

Category		Management	Likelihood of cancer
0	Need additional imaging or prior examinations	Recall for additional imaging and/or await prior examinations	n/a
1	Negative	Routine screening	Essentially 0%
2	Benign	Routine screening	Essentially 0%
3	Probably Benign	Short interval-follow-up (6 month) or continued	>0 % but ≤ 2%
4	Suspicious	Tissue diagnosis	4a. low suspicion for malignancy (>2% to ≤ 10%) 4b. moderate suspicion for malignancy (>10% to ≤ 50%) 4c. high suspicion for malignancy (>50% to <95%)
5	Highly suggestive of malignancy	Tissue diagnosis	≥95%
6	Known biopsy-proven	Surgical excision when clinical appropriate	n/a

Figure 2.1: BI-RADS Chart [9]

2.1.2 Pre-processing and Segmentation

Mammograms are characterized by noise and inconsistency of interpretation in their raw form. In order to bring forth a reliable representation of the breast anatomy, pre-processing techniques must be applied.
Additionally segmentation simplifies the raw image into a more meaningful and easier to analyze representation. Image segmentation is globally applied to locate objects and boundaries (lines, curves, etc.) in images. It also assigns a label to every pixel such that pixels having the same label share the same characteristics.

Cropping
Cropping can reduce the existing noise by eliminating artifacts like written labels or backgrounds.
In order to extract the regions of interest (ROIs) from the full raw images, Min Dong et al [26], Ramos et al [33] , Jiang et al [12] and Duraisamy et al[37] applied a manual cropping technique. Images were cropped into either 140x140 or 61x61 sub-images whose centers correspond to the truth-marked centers of lesions. For images without masses, the same procedure is followed, except that the locations were randomly taken from a healthy part of the mammogram. An average of 150 images were acquired.

Intensity adjustment
Dong et al [26] map the intensity value linearly to new values distributed in the gray level range [0255] by using brightness adjustment methods. The Rough Set (RS) theory was introduced to enhance the ROIs and it took into consideration the similarity of gray value between the gland tissues and the lesions which complicates the search for a suitable threshold. The theory also takes into account that the masses and their surroundings have a noticeable difference in gray levels and that the information can be reserved by increasing the edge gradient. Furthermore, the RS method is able to deal with uncertainty and vagueness using the indiscernibility relation concept.
[43] used the Histogram Equalization technique to reduce the over brightness or over darkness in the image. Histogram equalization is a method for adjusting image intensities in order to enhance contrast.
Duraisamy et al [37] utilized global contrast normalization method to normalize these patches. This technique computes the mean and standard deviation of all pixels. Afterwards, the mean of the intensities is subtracted from each pixel.
Mass lesion refinement was exploited by [30] and [12] since the mass lesions include both mass and breast tissues. To isolate the latter regions, Otsu algorithm is implemented. This method, used in computer vision and image processing, performs clustering-based image thresholding. It transforms the gray-level into a binary image. Foreground pixels and background pixels, following the bi-modal histogram, are separated by the algorithm. The optimum threshold separating the two classes is then calculated to achieve the inter-class maximal variance. Morphological operations are further applied to refine the mass region. Morphological operations is a technique relying only on the relative ordering of the values of pixels, instead of their numerical values, which makes it suitable for binary

images processing.

Multiresolution analysis
Image analysis and processing in multiple aspects allows image resolution to be changed for processing the smallest amount of data, as possible, by selecting relevant details for a given visual task. The basic idea of multiresolution analysis used by [33] is to produce sub-images from the original image, from coarse to fine resolution, and then analyze them in the frequency domain. In image-processing it is a powerful tool to preserve local information.

Wavelet Transform
Wavelet Transforms are widely applied in image processing and analysis, due to their accurate description of local structure (the breast anatomy as per our concern). Ramos et al [33] used the 2 dimensional discrete wave transform (2D-DWT) where the original image yields to four sub-band images that represents horizontal, vertical and diagonal directions.
The dual-tree complex wavelet transform (DT-CWT) has been used by Berks et al [24] because it offers a directionally selective representation with approximately shift-invariant coefficient magnitudes and local phase information. To form both real and imaginary parts of complex coefficients, DT-CWT gathers the results of two discrete transforms with a 90 phase difference. It provides six directional sub-bands with different angles which makes it easier to compute and richer in description than the monogenic signal used by Wai et al.
Similarly, Symlet transform was used by Usha et al [34]. This wavelet transform decomposes the image into constituent parts in the time-frequency domain. In his work, [34] uses four frequencies (approximation, horizontal detail, vertical detail and diagonal detail, respectively).

Ridgelet Transform
Wavelets deal with piecewise smooth functions in one dimension having only point singularities (Points with bad behavior, lesions for our case). Nevertheless, in spite of being extremely used in image processing in higher dimensions, wavelets are not powerful enough when other kinds of singularities exist. To overcome this problem, the ridgelet transform was created to handle 2-dimensional singularities efficiently. The main concept is to map a line singularity into a point singularity using the Radon transform. Ramos et al [33] applied the Ridgelet transform in its discrete version.

Active Contour Model
The mass segmentation procedure in [26] is realized by employing the active contour model (snake). The snake model consists of a parametric curve directed by internal and external forces that pull the curve toward the edge of the object until the energy function reaches the minimum. In computer vision, active contour model is a technique that delineate an object outline from a noisy 2D image. Snakes are popular in image segmentation and edge detection.

Data augmentation

Levy et al [6] studied the impact of data augmentation to increase the relatively small size of the training set. Small size is considered as characteristic of a lot of medical image datasets. Rotation, cropping, and mirroring transformations were used to increase the effective size of the DDSM dataset. 5 random rotations and 5 random crops per rotation sere applied to each training image. This method increased the training set size by a factor of 25. Random mirroring was also performed at training time. These augmentations are justified because the masses do not have any inherent orientation and their diagnosis is invariant to these kinds of transformations.

Jiang et al [12] added that a lesion must be recognized regardless of any particular orientation. Therefore, data augmentation helps increasing the network robustness. In this study, each breast lesion, after refinement, has another seven new samples generated by a combination of flipping and rotation (90', 180' and 270') transformations. Moreover, new samples were created by Duraisamy et al [37] artificially by applying several transformations to the original data in order to achieve a state-of-the-art performance, a large number of training images was needed to train the deep learning networks in Duraisamys model. Horizontally flipping, random crops and color jittering are all applied to limited number of input images. In this work, seven new label-preserving images were produced from each training image.

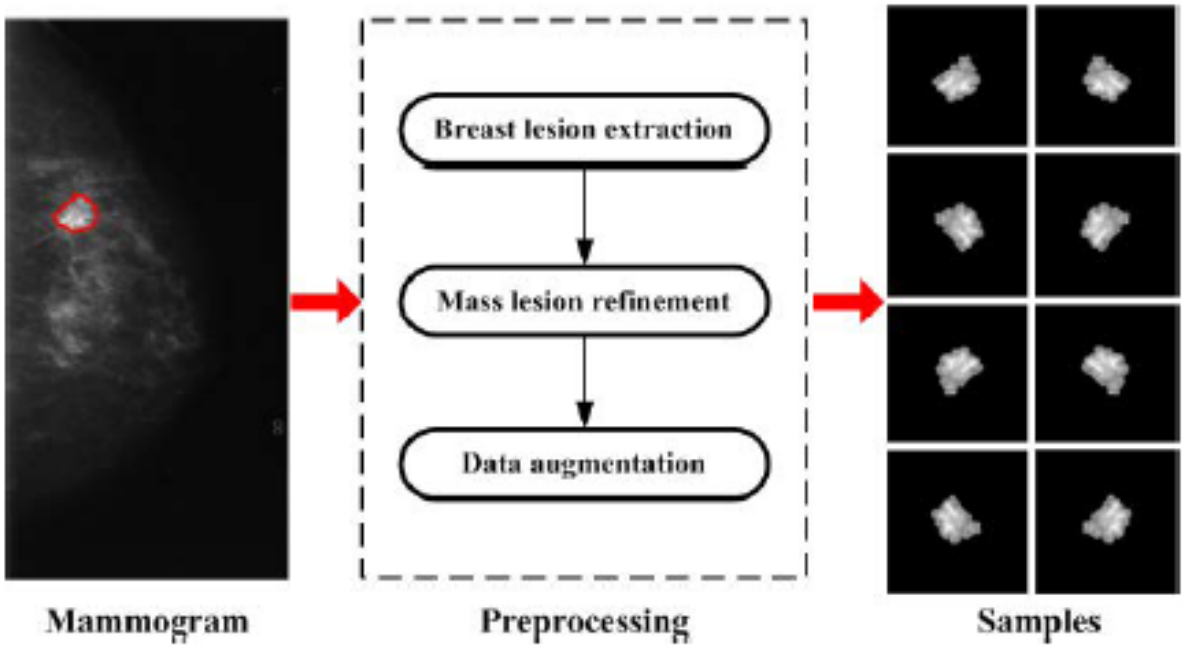


Figure 2.2: Workflow of data preprocessing [12]

2.1.3 Feature Extraction and Classification

The aim of the classification phase is to characterize each cluster as either malignant or benign using the selected features. Classification was defined by Vibha et al [43] as the segregation of the data into segments which do not overlap. The approach assumes some knowledge about the data known as Training. The classification accuracy measures the performance of the algorithm. The outcome of classification can then be represented as:
-True positive (TP): A tuple predicted to be in class, and is actually in it.
-False positive (FP): A tuple predicted to be in class, but is actually not in it.
-True negative (TN): A tuple not predicted to be in class, and is actually not in it.
-False negative (FN): A tuple not predicted to be in class, but is actually in it.
The precision and recall are used to determine the accuracy of the classifier.
Precision = TP/ (FP + TP)
Recall = TP/ (FN + TP)
The following approaches were taken by previous works.

Rule-based expert system
Papadoboulos et al [1] adopted the rule-based expert system approach. This method is used to store and manipulate knowledge in order to interpret information in an efficient way. It is often used in AI research and applications. Initially, for every pair of features, a 2D graph of the dataset in the two-feature space is implemented. This generates an average of 400 2-D plots. From the inspection of the 2D plots, four of those plots were found to have the highest discriminative value. Hence, in a two-feature space, a straight line (defining a linear decision rule) can be plotted which delimits a region (half space) containing a number of points that mostly belong (90 percent) to the same class.
The pairs of features corresponding to those plots of the MIAS database are:
-mean microcalcification cluster eccentricity (mean contrast)
-mean local background (mean distance from cluster centroid)
-mean contrast (number of microcalcifications in cluster)
-mean local background (standard deviation of the microcalcification distances from the cluster centroid).

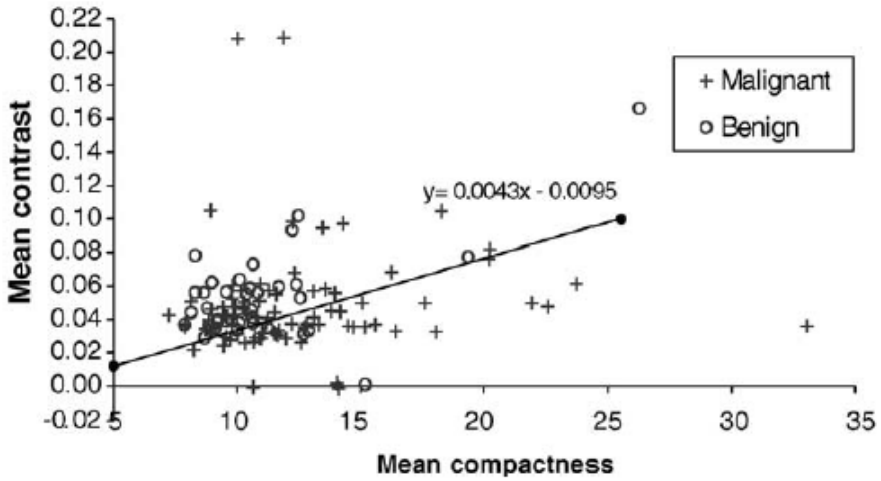


Figure 2.3: mean contrast vs. mean compactness and the corresponding linear decision boundary [1]

Neural classifier

Papadoboulos [1] used an artificial neural network (ANN) classifier. ANN is a feedforward multilayer perceptron linked with sigmoid hidden nodes. This architecture comprises one hidden layer with fifteen sigmoid nodes and an output layer containing one sigmoid node, whose value shows a malignant or a benign microcalcification cluster. Additionally, principal component analysis (PCA) has been applied to minimize the size of the input feature vector. The outcome of the PCA is a smaller feature vector formed of seven features. Subsequently, those features are normalized to zero mean and unit variance. Gradient decent, resilient backpropagation, conjugate gradient and quasi-Newton methodologies were added to the ANN training in order to choose the one with the best classification performance. Similarly, Al Hussein [27] implemented a convolutional neural network (CNN) architecture. Batch Normalization was applied after the first two convolutional layers. The output vector is flattened into a 1D vector, after the feature extraction, then the output is fed to the fully connected layers. Furthermore, Duraisamy et al [37] applied a FCRN classifier. A fully complex-valued single hidden layer neural network is used with a Gaussian and an exponential activation function. A projection-based learning algorithm is employed to predict the network parameters. The optimal output weights values are estimated by this algorithm as a result of a linear equations system. By means of circular transformation, real-valued input features are mapped to the complex domain. Moreover, [37] applied Deep Learning convolutional neural network (DL-CNN) in addition to the FCRN. The architecture is applied in a patch-wise manner. Large patches capture the required local information in order to correctly classify pixels belonging to different breast tissues. DL-CNN FCRN consists of three stages of convolutional layers, ReLU activation layers, and max pooling layers, followed by three fully connected layers.

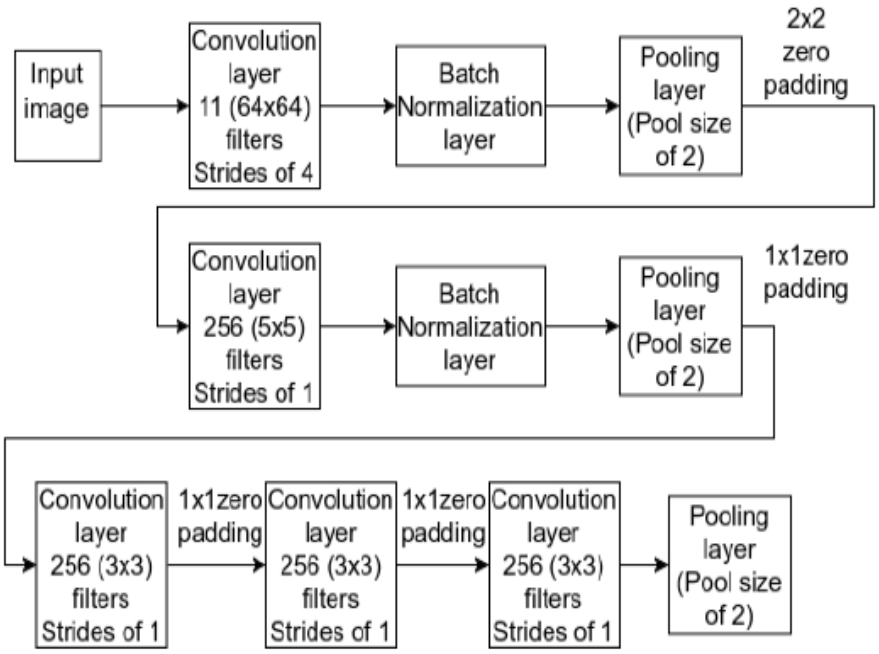


Figure 2.4: single CNN architecture by [27]

Candidate Generation with m-DBN and GMM was explored by Dhungel et al [30]. In machine learning, a multiple deep belief network (m-DBN) is a graphical model class of deep neural network. It comprises multiple layers of latent units (hidden variables), where layers are connected but units within each layer are independent. In statistics, a Gaussian mixture model (GMM) is a probabilistic technique for subpopulations representations within a population. Some implementation methods of mixture models involve attributing postulated sub-population-identities to individual observations, which can be interpreted as types of unsupervised learning or clustering procedures. In Dhungel work, the m-DBN classifier is trained to detect candidates in a mammogram using a grid search at four different resolutions, breast-air boundary segmentation (using Otsu's segmentation) is used as a mask. Essentially, the m-DBN binary classifiers have positive or negative labels. The coarsest resolution image having samples that are classified as positive are used to train the next resolution. Hence, multi-scale cascade of DBN classifier is formed. The inference runs in all positions of the grid (i.e., every discrete position existing within the breast mask of the corresponding image resolution). The mean field approximation of the values in all DBN layers is followed by the calculation of the free energy on the top layer. Besides m-DBN, pixel-wise GMM model is used on the full resolution image. Then, the results of m-DBN and GMM are combined with CCA (connected component analysis), using a similarity measure. The output from CCA consists of clusters of pixels which are classified as belonging to a breast lesion.

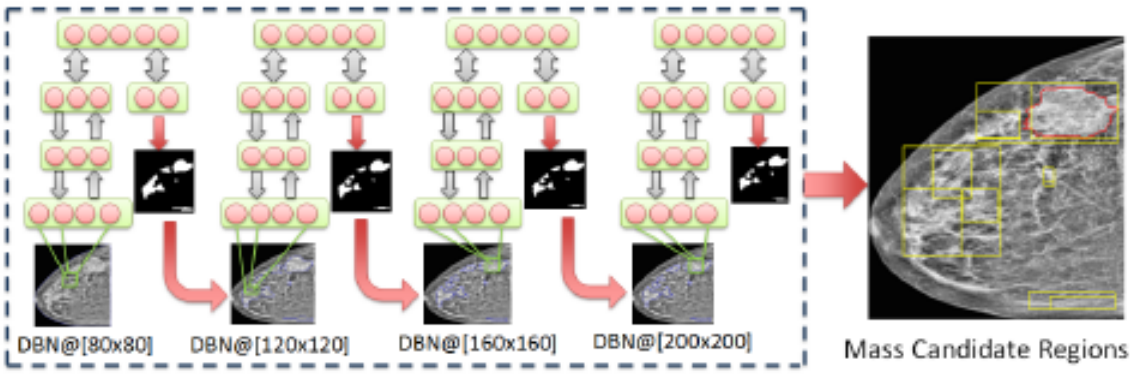


Figure 2.5: DBN layers [30]

Faster R-CNN contains a set of convolutional layers, known as Region Proposal Network (RPN), on top of the last convolutional layer of the main network, which is trained to detect and localize objects on the image, regardless of the class of the object. It creates detection boxes with different sizes and aspect ratios in order to find objects with different sizes and shapes. The top scoring default boxes are called region proposals. The neural network evaluates the signal of each proposed region of the previous convolutional layer. Both branches solve a classification task independently to find objects. The best predictions are chosen from the detected overlapping objects according to the non-maximum suppression technique.

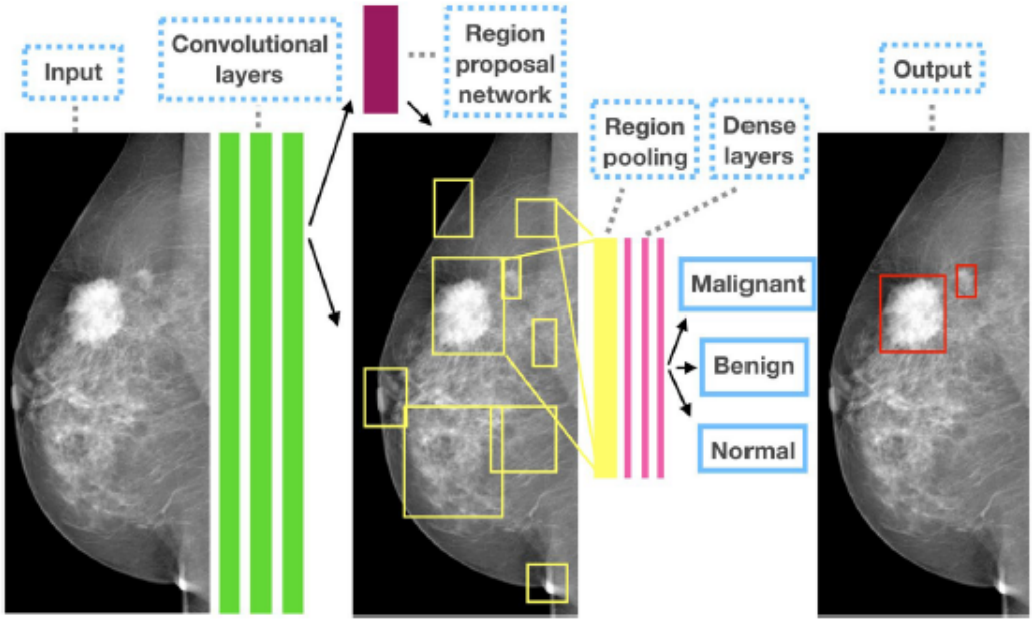


Figure 2.6: The outline of the Faster R-CNN model for CAD [8]

Transfer learning

Transfer learning is a machine learning technique that depends on storing knowledge gained while solving one problem and applying it to a different but related problem.
Levy et al [6] evaluated three network architectures: a baseline CNN, an AlexNet and a GoogLeNet. They used the same main network as the original works but replaced the last fully-connected (FC) layer to output 2 classes. They also added batch normalization (improving the performance and stability of artificial neural networks to normalize the input layer by adjusting and scaling the activations). A 224x224x3 image is taken as input. It comprises three convolutional blocks composed of 3x3 Convolutions, Batch Normalization, ReLU (a rectifier) and Max Pooling (a sample-based discretization process), with respectively 32, 32 and 64 filters each. A three FC layers of size 128, 64 and 2 follows. The final layer is a soft-max layer responsible for binary classification. Levy implemented ReLU activation functions, Xavier weight initialization, and the Adam update rule (a gradient descent optimizer).
Jiang et al [12] also took advantage of transfer learning by pre-training the model on ImageNet. They tuned the parameters with samples from the BCDR database and transferred these learned generic features on ImageNet for mammogram-based breast cancer diagnosis.
VGG16 network (shown in figure 2.7) is a CNN model introduced by Simonyan and Zisserman in the paper Very Deep Convolutional Networks for Large-Scale Image Recognition. The model achieves 92.7 percent top-5 test accuracy in ImageNet and AlexNet. Ribli et al [8] used this algorithm in their work. Accordingly, the final layer of the network is able to detect two kinds of objects in the images, benign or malignant masses. The technique outputs a bounding box for each detected lesion and a score representing the confidence in the class of the lesion. The maximum of the scores of all present malignant lesions detected in the image is used to assign a final score to the mammogram.

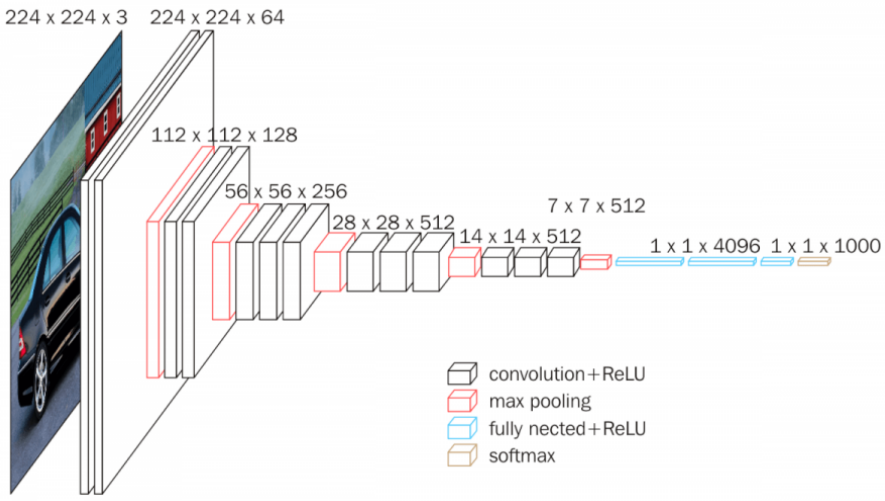


Figure 2.7: VGG16 network

False Positive Reduction is required since a significant amount of false positives is present. False positive is usually two orders of magnitude greater than the number of true positives. The previous algorithms may suggest possible locations of lesions which are incorrect. Fully complex-valued relaxation network classifiers are used with the previous models in the last stage of CNN networks. The relaxation process is a useful technique for reducing local ambiguity and reach global consistency, hence reducing the false positives.

K-means Clustering
k-means clustering is a technique used for vector quantization. It aims to partition n observations into k clusters by assigning each observation to the cluster with the nearest mean. This method was used by Elmoufidi et al [2] to detect the ROIs in mammograms.

Support Vector Machines
In their work, Papadopoulos et al [1] and Dong et al [26] used Support Vector Machine (SVM). SVM is a supervised learning model that analyzes data used for classification. It is based on the Vapnik-Chervonenkis (VC) and structural risk minimization (SRM) principle. It performs the best generalization ability by finding an optimal separating hyper-plane and increasing the margin between classes.

Dhungel et al [30] combined CNN and SVM in one architecture. This model is known as **R-CNN** in computer vision literature. In a similar manner, Ribli et al [8] used a Faster R-CNN model. R-CNN is based on a convolutional neural network in addition to several components for detecting, localizing and classifying the different objects in an image.
Faster R-CNN contains a set of convolutional layers, known as Region Proposal Network (RPN), on top of the last convolutional layer of the main network, which is trained to detect and localize objects on the image, regardless of the class of the object. It creates detection boxes with different sizes and aspect ratios in order to find objects with different sizes and shapes. The top scoring default boxes are called region proposals. The neural network evaluates the signal of each proposed region of the previous convolutional layer. Both branches solve a classification task independently to find objects. The best predictions are chosen from the detected overlapping objects according to the non-maximum suppression technique.

Random Forest
Decision trees are characterized by overfitting their training set. Usha [34], Vibha [43], Dhungel [30], Liu et al [18] and Berks et al [24] used Random Forest to solve this problem faced by Dong et al [26].
Random forest (RF) is a composite classifier built from a collection of independent trees. This ensemble learning method is used for classification and regression tasks. RF constructs a series of decision trees at training time. The output is the mode class for classification or the mean estimation for regression problems. RF are widely used because of their rapid training, resistance to overfitting, and near state-of-the-art performance.
[43] construted each tree in four steps as follows:

Step 1: Using bootstrap, Random sampling procedure outputs the data for the training set, where two-thirds of the data is used for creating the tree.
Step 2: A random set of attributes is chosen from the previous feature set. The ones with most information gain is identified to adjust each node.
Step 3: Keep adding nodes to the tree until the information loss threshold is reached.
Step 4: Define the error estimate by running the remaining one-third of the dataset and calculate the correctness.
Dhungel et al [30] proposed a classification process based on random forest in order to extract texture and morphological features from the mammograms. These features include object-based measures, including the number of perimeter pixels, area, perimeter-to-area ratio, circularity, rectangularity, and normalized radial length (NRL). The NRL features include: mean value, standard deviation, entropy, area ratio and zero-crossing count. Moreover, gray level co-occurrence matrix (GLCM) extracts the texture features. Hence, the output is used in the training and inference processes of a double-stage cascade random forest classifier.
Similarly, Ramos et al [33] implemented the random forest algorithm for ROIs classification stage as a data mining method to separate the data into non-overlapping segments. According to Berks et al [24] approach, given N samples of the training set, each of which is a D-dimensional feature vector with a label of one of C classes, a random forest includes a set of tree predictors built from the training data. Each tree in the forest is constructed from a bootstrap sample of the training set (a set of N samples selected randomly from the original dataset). The forest is constructed using a standard classification and regression tree (CART) algorithm. Nonetheless, instead of assessing all D dimensions for the optimal split at every tree node, only a random subset of d ¡ D dimensions are taken into account. Each tree casts only one class vote, and the class having the greatest number of votes is assigned to the input vector. Alternatively, the untaken votes accredited to each class can be used to supply a probabilistic labelling of the input vector.
In conclusion, random forests are suitable for learning non-linear relationships in high-dimensional multi-class training datasets. They have been proven to perform as well as CNN or support vector machines, whilst being more efficient to compute.

2.1.4 Evaluation metrics

The objective evaluation criterion used in the previously mentioned papers is mainly assessed by the classification sensitivity, specificity, accuracy, positive predictive value (PPV), negative predictive value (NPV), Matthews correlation coefficient (MCC), and receiver operating characteristic (ROC).
Below are the main calculations of each criterion as explained by Dong et al [26].
Sensitivity is concerned only with positive cases; it shows the part of the detected positive cases over the actual positive cases.
Sensitivity = TP / (TP + FN)
Specificity deals only with negative cases; it shows the proportion of the detected negative

cases over the actual negative cases.
Specificity = TN / (TN + FP)
PPV handles all cases; it can be measured as the correct detected positive cases over all detected positive cases.
PPV = TP / (TP + FP)
NPV manages all cases; it can be computed as the correct detected negative cases over all detected negative cases.
NPV = TN / (TN + FN)
Accuracy includes all cases, and it is the most frequently calculated indicator; it shows the precision of the predicted results.
Accuracy = (TP + TN) / (TP + TN + FP + FN)
MCC is a powerful accuracy evaluation indicator for machine learning models. It is mostly effective for the case of the unbalanced negative sample numbers compared with the positive sample numbers.

Additionally, the receiver operating characteristic (ROC) curve measures the predictive accuracy of the proposed approach. It indicates the true and false positive rate. The AUC which is the area under the ROC curve is one of the best representation techniques for comparing classifiers in two-class issues. The test results are known to perform better if the ROC curve rises quickly toward the upper left corner of the graph or if a large AUC value is achieved. On one hand, a 1.0 indicates that the diagnostic test is reliable; on the other hand, an area close to 0.5 demonstrates that more enhancements need to be performed.

To validate the results, Dong et al [26] used the fivefold Cross Validation (5-CV) to evaluate the classification accuracy, where the feature data (positive 115, negative 85) are randomly split into five categories, every category is chosen at a time as testing set and the remaining four categories run as the training set. By performing five loops, the average classification accuracy can be computed. By using this method, all of the test sets are guaranteed to be independent, thus the liability of the output is improved.

2.1.5 Discussion

Papadopoulos et al [1] evaluated their methodology on the MIAS datasets. They correctly characterized 11 (61 percent) malignant clusters and the false classification of 1 (7.6 percent) benign. The above results are quite low and thus a rule based classifier could not provide useful diagnostic information, by both [1] and [43]. In the final stage ANNs and SVMs have been used. SVMs using a Gaussian kernel function provided the best performance (0.83 classification rate and Az = 0.81).

Levy et al [6] proposed an end-to-end deep learning model and showed how preprocessing, data augmentation and transfer learning can surpass the data bottleneck commonly found in medical computer vision tasks. The best model achieved 0.934 recall at 0.924 precision, which outperforms trained radiologists and reaches state-of-the-art. Similarly, Jiang et al [12] investigated transfer learning methods which improved the

prediction performance when compared to shallow neural networks and traditional techniques. Transfer learning was also proved to break the insufficient data samples problem for training deep neural networks. GoogleNet NN scored an AUC of 0.88 while AlexNet had an AUC of 0.83.

In [27], an accuracy of 80.1 percent was achieved and the AUC scored 0.78.
Duraisamy et al [37] achieved a high classification accuracy of 99 percent and sensitivity 0.9875 by using DL-CNN feature learning with FCRN classifier. The paper state that the proposed system is 1.0 which means that the system can detect all abnormalities without any false positives. Nevertheless, several research groups employed manually identified cluster ROIs to train and validate their works. This approach resulted in more subjective detections and performance (0.9 accuracy and higher), while the radiologists interpretation performance range from 0.54 to 0.72 for attendants and corresponds to 0.53 to 0.66 for residents. Therefore, we cannot fully rely on these approaches.

Random Forest Decision Classifier was employed by Vibha et al [43]. Using n-fold cross validation technique, experimental results have proved to be effective on the MIAS database. An accuracy of 86 - 90 percent was achieved. Also, Berks et al [24] achieved a state-of the art performance using random forest to detect and classify curvilinear structures in mammograms. Other models also reached good results too as shown in figure 2.8
Multiresolution techniques represented powerful tool for image processing in Ramos et al [33] work, due to its resemblance to the human visual system by selecting relevant details to execute visual recognition tasks. In this work, random forest gave an AUC = 0.9.

Line detection algorithm	ROC A_z	Sensitivity for fixed 90% specificity	Mean Absolute Error of orientation
DT-CWT/RF, $w = 1$, $s = 5$	0.923±0.00036	0.792	15.88
Linop/RF, $w = 1$, $s = 5$, 8 orientations	0.911±0.00048	0.757	19.35
Gaussian/RF, $w = 1$, $s = 4$, $\sigma_{min} = 1$	0.901±0.00048	0.731	21.37
Monogenic/RF, $w = 1$, $s = 4$, $\lambda = 4$	0.868±0.00045	0.643	33.73
Monogenic, $s = 3$, $\lambda = 4$	0.818±0.00055	0.547	30.86
Gaussian, $s = 4$, $\sigma_{min} = 1$	0.813±0.00053	0.533	24.27
Linop, $s = 5$, 8 orientations	0.737±0.00060	0.413	29.32

Figure 2.8: Line detection and orientation computation results [24]

The Faster R-CNN proposed by Ribli et al [8], showed the highest performance among all the previously mentioned works. It achieved second place in the Digital Mammography DREAM Challenge with an AUC = 0.85 score. This work detected lesions locations and not just classified the image. The system detects 90 percent of the malignant lesions with

only 0.3 false positive marks per image in the INbreast dataset. However, it is crucial to note that the outputs obtained on different datasets cannot be directly comparable.

2.2 Concepts overview

2.2.1 Mammography

Mammography (also called mastography) is the process of getting an x-ray picture of the human breast. It is used for breast cancer diagnosis and screening, mostly in women who have either no signs or symptoms of breast cancer or have a lump or other sign of the disease. Mammography also provide early detection of breast cancer by localizing masses and microcalcifications.
Mammogram images are created by applying doses of ionizing radiation. Generally, two views of each breast (right and left) are acquired. The first view is a cranial-caudal view, referred to as CC; it is a top-down view of the breast. The second view is a mediolateral-oblique, reffered to as MLO; it is a view from a side with an angle as shown in figure 2.10.
Abnormal findings are then analyzed. We mainly study two of the findings: breast masses and microcalcifications. Calcium are deposited forming the calcifications while lumps and tumors constitute the masses. However, these findings are not necessarily cancerous. They can either be benign or malignant (cancerous).

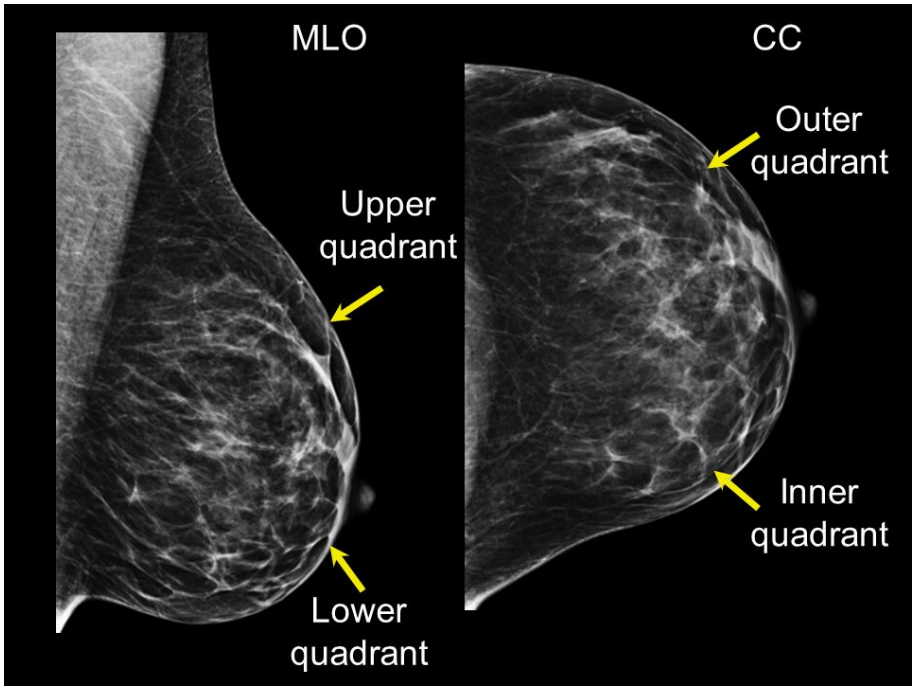


Figure 2.9: MLO and CC views by Breast Imaging Medical School University of Ottawa

2.2.2 Machine Learning

Machine learning (ML) is the study of algorithms and statistical models in order to perform a certain task in absence of explicit instructions. It is a part of artificial intelligence. ML is then able to make predictions or decisions for the given task. It is widely used in computer vision and data mining as it focuses on exploratory data analysis.

2.2.3 Computer Vision

Computer vision is a field that attempts to automate tasks that can be done by the human visual system in computers. These tasks include techniques for acquiring, processing, analyzing and understanding digital images in order to form numerical information. It transforms the visual images into descriptions of the world to trigger a certain action or decision. This image understanding is basically supported by geometry, physics, statistics, and learning theory.

2.2.4 Classification

Classification is the method of categorizing a new observation (instance) to its belonging set. Assigning a diagnosis, for example, to a certain patient based on observed characteristics of the patient (sex, blood pressure, certain symptoms, etc.) is an application in machine learning. Classification is considered to be an instance of supervised learning where the training set is present along with the correct observations.

2.2.5 Ensemble Learning

In statistics and machine learning, ensemble methods are a way of using multiple learning algorithms in order to enhance the predictive performance obtained from any of the individual models. An ensemble is a supervised machine learning algorithm that represents a single hypothesis. Thus, ensembles are proved to have more adaptability and flexibility in the tasks they can represent. They are able to over-fit the training data more than a single model would. However, other ensemble methods, such as bagging, tend to reduce the over-fitting problems of the training dataset. In addition, ensembles tend to yield better results when the models are diverse. On the other hand, some random algorithms, such as random decision trees, are able to produce a stronger ensemble than other deliberate algorithms.

2.2.6 Decision Trees

Decision tree learning is a predictive model that transforms observations about an item (the branches) into conclusions about its target value (the leaves). It is a simple form of binary trees. Tree models where the target variable is represented as a discrete set of values are called classification trees. Leaves, in this previous structure, represent class labels while branches represent conjunctions of features.

2.2.7 Random Forest

In ensemble learning, Random forests (RF) or random decision forests operates by constructing several decision trees at training time and merges them together to get a more accurate and stable prediction. The mode class is then selected as the output for classification tasks, while the mean prediction is selected for regression problems. RF have almost the same hyperparameters as a decision tree or a bagging classifier.
Random decision forests help in improving the overfitting habits of training sets. This is achieved by adding randomness to the model. Rather than searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. Addition randomness can be added by assigning random thresholds for each feature instead of searching for the best possible thresholds.

Chapter 3

Methodology

3.1 Dataset

Four choices were available for the dataset. The first dataset was the Digital Database for Screening Mammography (DDSM). The database contains a set of 2,500 studies. Two images of each breast are included per study and each breast consists of a mediolateral oblique and craniocaudal view mammogram. In computer vision, these multi-view images are mostly used in finding similarities and recognizing patterns which is not the aim of this thesis. The second dataset was the Inbreast dataset. Inbreast contains studies of mastectomy patients. Mastectomy is a breast cancer surgery that requires removing the entire breast, chest wall, and all axillary lymph nodes. Usually, after the surgery, many side effects can develop including blood clots, wound infections, seroma (Fluid Collection) and haematoma (Blood Collection). Therefore, these mammograms can increase the false positive rate and are excluded from the initial dataset selection. The third dataset was the Breast Cancer Digital Repository (BCDR). The BCDR is mainly used in cancer diagnosis rather than cancer detection as it uses the Bi-Rad notation. Thus, the mini-MIAS dataset was selected as the initial training and testing dataset.

Exploring the MIAS dataset

The Mammographic Image Analysis Society is an organization located in the UK. The society collected 322 digitized images from the UK National Breast Screening Programme, 1024 1024 pixels each. Truth-markings for locations of present abnormalities are attached. The dataset is widely covered in many papers ([1], [26], [43], [37]) due to its public and free availability via the Pilot European Image Processing Archive (PEIPA) at the University of Essex.
The raw data is represented in figures 3.1 and 3.2 .

```
In [8]: plt.imshow(read_pgm(glob('C:/Users/victory/Desktop/MIAS/*.pgm')[0]))
Out[8]: <matplotlib.image.AxesImage at 0x1fe033037b8>
```

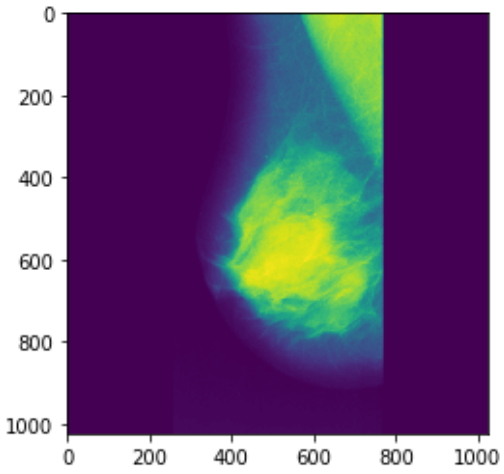


Figure 3.1: MIAS raw image (left breast)

```
In [9]: plt.imshow(read_pgm(glob('C:/Users/victory/Desktop/MIAS/*.pgm')[1]))
Out[9]: <matplotlib.image.AxesImage at 0x1fe0320d978>
```

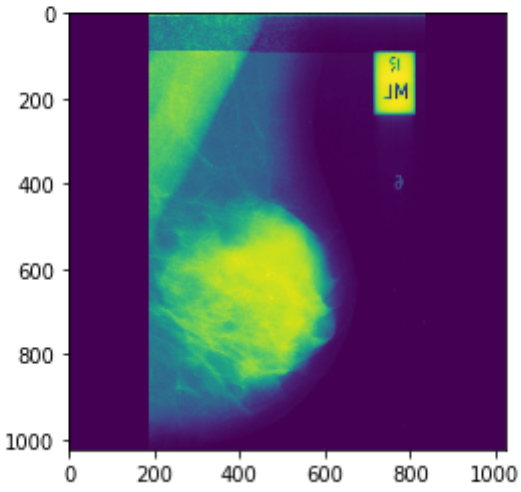


Figure 3.2: MIAS raw image (right breast)

The MIAS dataset contains a csv file which comprises all the relevant information as shown in figure 3.3 .

```
In [2]: #Display Dataframe
        all_cases_df = pd.read_table('C:/Users/victory/Desktop/MIAS/mammogram_info.csv', delimiter=' ')
        all_cases_df.head(25)
```

Out[2]:

	Reference	Bkg	Class	Sever	X	Y	Rad
0	mdb001	G	CIRC	B	535	425	197.0
1	mdb002	G	CIRC	B	522	280	69.0
2	mdb003	D	NORM	NaN	NaN	NaN	NaN
3	mdb004	D	NORM	NaN	NaN	NaN	NaN
4	mdb005	F	CIRC	B	477	133	30.0
5	mdb005	F	CIRC	B	500	168	26.0
6	mdb006	F	NORM	NaN	NaN	NaN	NaN
7	mdb007	G	NORM	NaN	NaN	NaN	NaN
8	mdb008	G	NORM	NaN	NaN	NaN	NaN
9	mdb009	F	NORM	NaN	NaN	NaN	NaN
10	mdb010	F	CIRC	B	525	425	33.0
11	mdb011	F	NORM	NaN	NaN	NaN	NaN
12	mdb012	F	CIRC	B	471	458	40.0
13	mdb013	G	MISC	B	667	365	31.0

Figure 3.3: MIAS information table

The CSV file contains 7 columns as follows;
The first column is the MIAS database reference number.

The second column represents the character of background tissue.
F Fatty
G Fatty-glandular
D Dense-glandular

The third column contains the present class of abnormality:
CALC Calcification
CIRC Well-defined/circumscribed masses
SPIC Spiculated masses
MISC Other, ill-defined masses
ARCH Architectural distortion
ASYM Asymmetry
NORM Normal

The fourth column represents the severity of abnormality;
B Benign
M Malignant

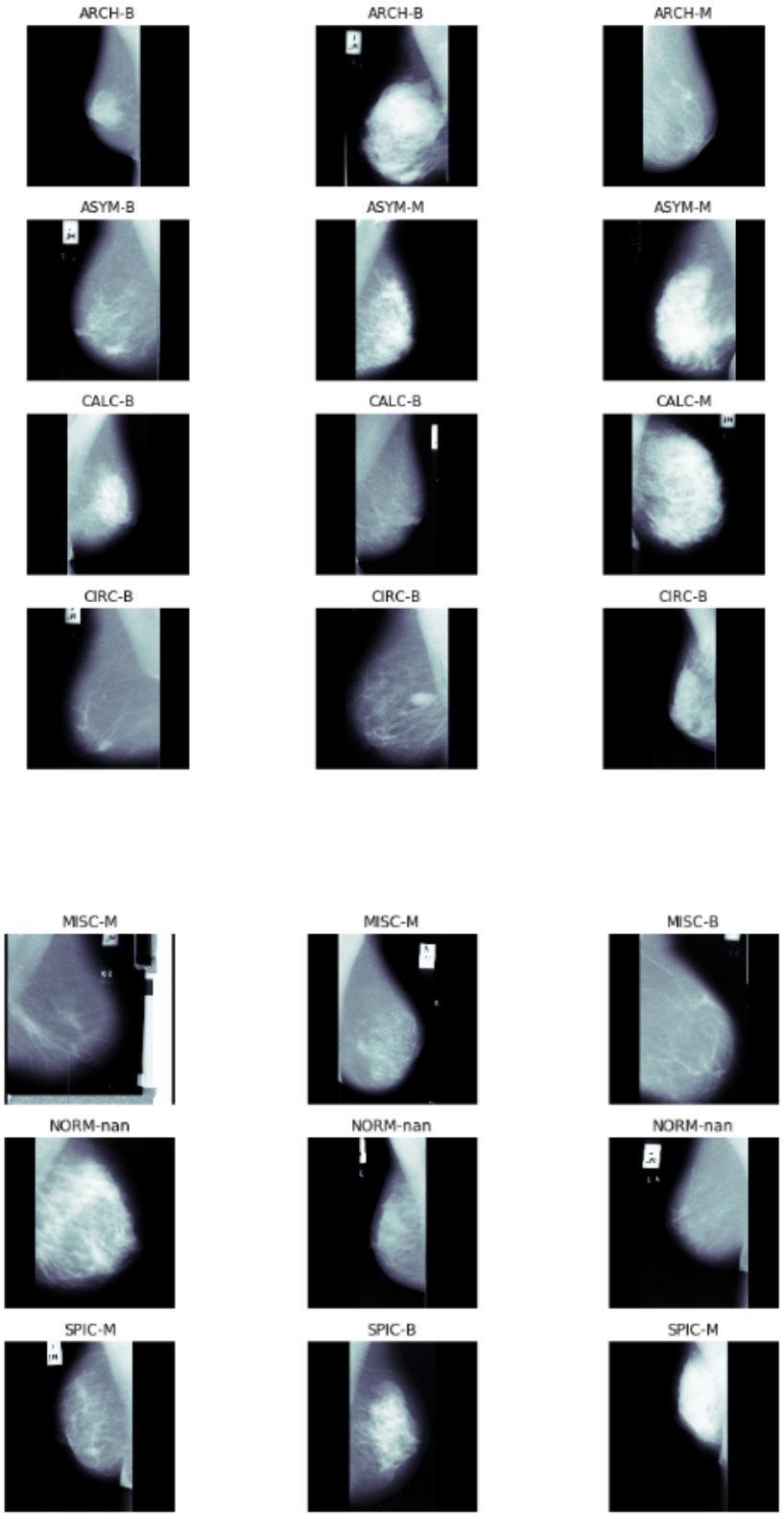


Figure 3.4: An overview of the different abnormality classes and severity generated by the bachelor code

The fifth and sixth columns are the x and y image-coordinates of the center of abnormality.

The seventh column represents the approximate radius (in pixels) of a circle enclosing the abnormality.

These pieces of information served in the pre-processing and classification phases. Each abnormality class and background tissue has different properties between shape, texture and intensity. These properties are handled in details in the pre-processing subsection. In addition, the image coordinates and radii of the abnormalities were used in the extraction of the regions of interest (ROI). Moreover, the severity of abnormality was considered as the label for the classification phase.

3.2 Pre-processing

3.2.1 Contrast Limited Adaptive Histogram Equalization (CLAHE)

The concept of histogram equalization[32] is mainly used to adjust the contrast in image processing. The global contrast of the image is usually increased by this technique, especially when close contrast values are present in the image. The intensities are adjusted by being better distributed on the image histogram. In particular, histogram equalization is useful in images with backgrounds and foregrounds and can lead to a clearer view of the breast anatomy in our case.

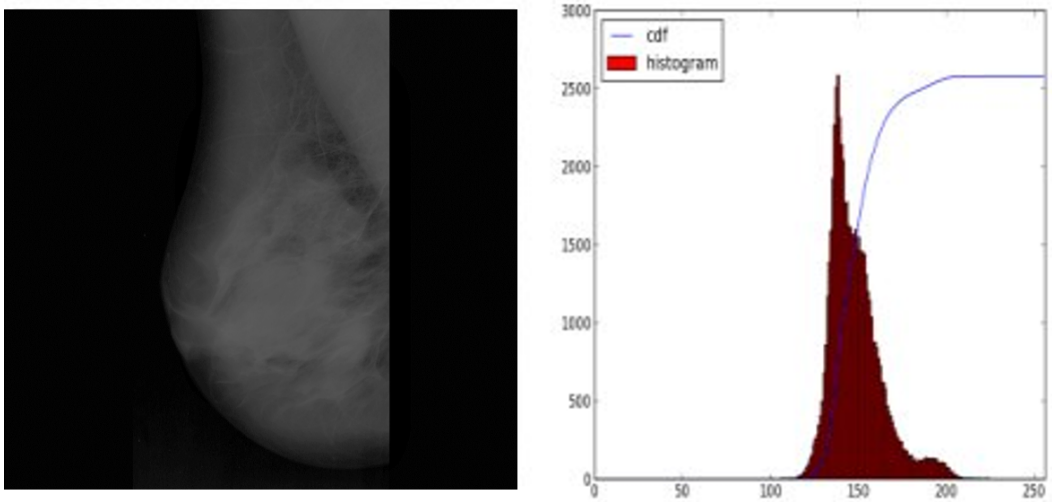


Figure 3.5: Histogram Equalization (before)

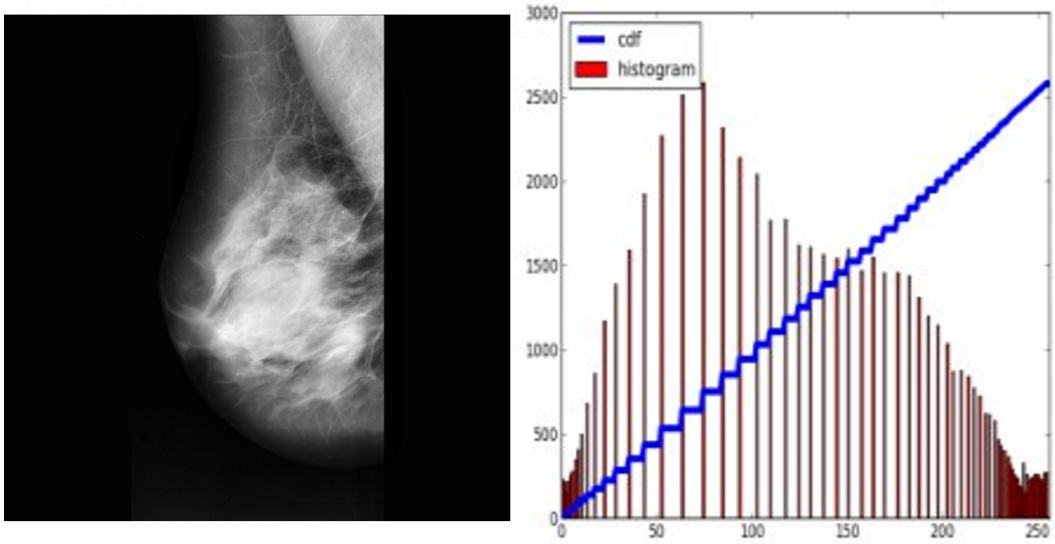


Figure 3.6: Histogram Equalization (after)

However, histogram equalization is ineffective when a large intensity variation exists (when the histogram covers a large intensity region). This is the case in all mammograms where the background has low intensity values while the glandes and masses have high intensity values.
Histogram equalization considers the global contrast of the image. Hence, most of the information is lost due to over-brightness. In order to solve this problem, we use adaptive histogram equalization where the image is partitioned into blocks (known as tiles). By default, the tile size is 8x8 in OpenCV. Then each tile is histogram equalized. Another problem arise when noise exist; it becomes amplified. Contrast limiting is applied to avoid the later problem. If any tiles histogram bin grows above a certain contrast limit (40 by default in OpenCV), the tile pixels are clipped and distributed uniformly to other bins before applying histogram equalization. Then, to remove any unwanted artifacts in tile borders, bilinear interpolation is applied.
Bilinear Interpolation [4] is a re-sampling algorithm that computes the distance weighted average of the four nearest neighbors pixel values in order to estimate a new pixel value. Pisano et al [11] were the main reference used in this phase as they proved that CLAHE can improve the detection of spiculations on dense mammographic backgrounds, which is the hardest background type that can be detected and diagnosed by radiologists.

3.2.2 Adaptive Mean Thresholding

Simple Thresholding is a method that checks if a certain pixel value is greater than a threshold value, it is assigned to a foreground value (white), else it is assigned to a background value (black).
By applying simple thresholding, we use a global value as the threshold value. However, if the image has varying illumination conditions in different areas (e.g. images occurring

as a result of a strong illumination gradient or shadows), adaptive thresholding is applied. The algorithm changes the threshold dynamically over the image.
Adaptive thresholding uses a grayscale or color image as input, then outputs a binary image representing the resulting segmentation as shown in figure 3.7 .

There exist three primary approaches to finding the threshold. The first approach is the Chow and Kaneko algorithm. It divides the image into a matrix of overlapping subimages and then calculates the optimum threshold for each subimage according to its histogram. The threshold for each single pixel is computed by interpolating the resulting outputs of the subimages. The drawback of this method is that it is computational expensive and, therefore, we apply a local threshold in this thesis. This alternative statistically examines the intensity values of the local neighborhood of each pixel and calculates the mean value. The size of the neighborhood must be sufficiently large in order to cover enough foreground and background pixels, otherwise threshold is not accurate. This is why the mean adaptive threshold is applied before cropping the mammograms to extract the regions ROIs. This method is less expensive than the Chow and Kaneko approach, especially with large datasets.
The preprocessing output is shown in figure 3.8 .

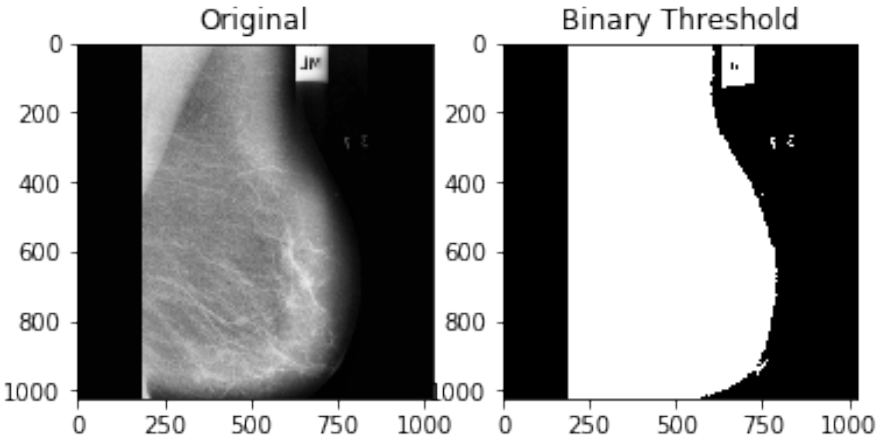


Figure 3.7: Thresholding

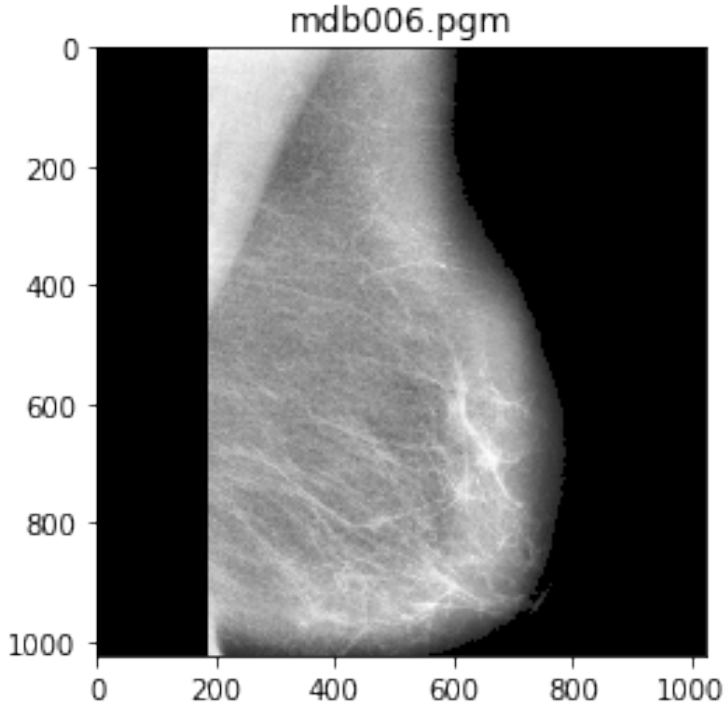


Figure 3.8: Pre-processed mammogram of the image in Figure 3.7

3.3 Regions of Interest (ROIs)

After the pre-processing phase, the normal mammogram background will be black, and the breast will show up in gray and white regions. Tissue that is denser, including connective tissue and glands, are white. Some breasts are denser which makes the detection of abnormalities harder because tumors are also made up of dense tissue and will appear white as well. The appearance of healthy mammograms can still vary. Additionally, the breast area in our dataset represents only 30 percent of the mammogram [39].So, in order to reduce the false positive rate, regions of interest (ROIs) are cropped from the pre-processed images with a 1:1 ratio so that the image preserves its features when resized (224x224 pixels) for the neural network input. The cropped image contains the lesion for the abnormal cases according to the CSV file attached with the dataset. For the normal cases, random crops are selected.

3.4 Feature Extraction

Feature extraction is a process in pattern recognition and image processing that begins with a set of raw measured data and derives some informative and non-redundant values (known as features). The resulting features are more manageable for processing and

human interpretations, while keeping the original description of the data in an accurate manner.
The input data is then transformed into a reduced feature vector without any redundant representation. Thus, the amount of memory and computation power are minimized. Also, the classification phase is more generalized and less vulnerable to overfitting the training samples.
The latter process can include low-level features (ex: edge, corner, curvature and changing intensity detection), shape-based features (ex: thresholding and template matching) or flexible methods (ex: active contours (snakes)).

In this thesis, transfer learning was used as a feature extraction technique. Transfer learning represents a technique for predictive modeling on a different but somehow similar task. It can then be reused partly or entirely in order to accelerate the training process and improve the performance of the model. This means that we are able to reuse the weights in one or more layers from a pre-trained network model in our new model. The weights can either be kept fixed, fine-tuned, or adapted to the new model of interest.

Typically in deep learning, the gathering of huge amounts of images is required. Images must also be classified or annotated before being fed into the network for training. To avoid this problem, Apache MXNet (an open-source deep learning software framework) contains several pre-trained deep learning models that were trained on the ImageNet database (an image database in which each node of the hierarchy contains an average of five hundred images). We can also use the pre-trained models in MXNet to extract a feature vector (a list of 2048 floating point values) of the models internal representation of a category.
Four of MXNet pre-trained models were used in this thesis.

3.4.1 ResNet

ResNet is one of the pre-trained models available in MXNet. ResNet is a short term for Residual Network.

A residual neural network (ResNet) [19] is an artificial neural network (ANN) that utilizes skip connections, or short-cuts to jump over some layers. An additional weight matrix may be used in order to learn the skip weights.
Generally, each layer directly feeds into the next layer in traditional neural networks. While using residual blocks, each layer feeds into the next layer and also into the layers about 2 to 3 hops away.
According to the universal approximation theorem, neural networks are universal function approximators and that when the number of layers increases, the accuracy increases and the network can learn both simplex and complex functions. However, problems like vanishing gradients (as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitively small) and curse of dimensionality may arise, and therefore the network becomes unable to learn simple functions like identity functions. In addition, as networks gets deeper, the training becomes difficult and the

accuracy begins to saturate and then degrades. This happens because the shallower networks learn better than their deeper counterparts.
Considering a neural network block, whose input is x and who learns the distribution H(x). Let us denote the difference (or the residual) between this as
R(x) = Output Input = H(x) x
By rearranging the previous function, we get
H(x) = R(x) + x
Thus, the layers in a traditional network are learning the true output (H(x)) whereas the layers in a residual network are learning the residual (R(x)).

In ResNet, identity shortcut connection is introduced. It skips one or more layers, as shown in the following figure.

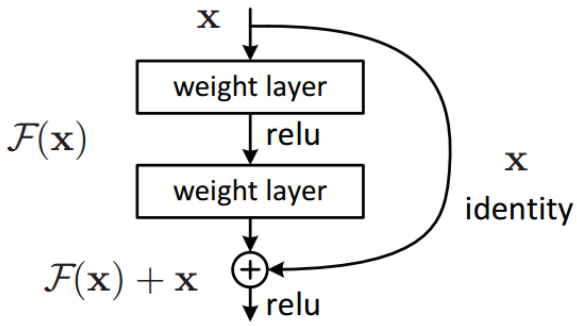


Figure 3.9: ResNet [19]

We reuse activations from a previous layer until the adjacent layer learns its weights. While training the network, the weights adapt to mute the upstream layer, and amplify the previously-skipped layer. The learning is therefore faster since there are fewer layers to propagate through. The network then gradually restores the skipped layers as it learns the feature space.

Two models of the ResNet were used in this thesis: **ResNet-50** (figure 3.10) and **ResNet-152**.
ResNet50 is a 50 layer Residual Network while ResNet152 is a 152 layer Residual Network.

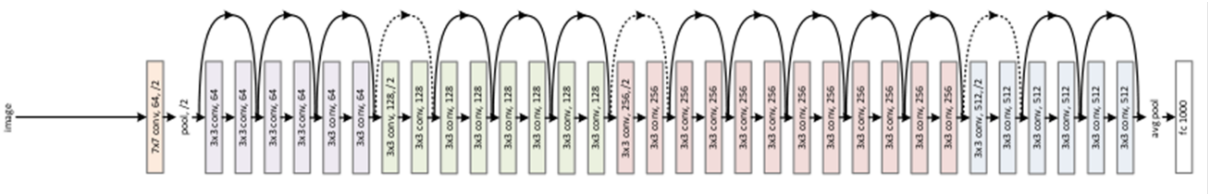


Figure 3.10: ResNet50 [19]

Both architectures mainly contain a variation of Batch Normalization, Convolution, Pooling and Fully-Connected layers in a specific setup.

The convolution layer (ConvNet) is a Deep Learning algorithm which assigns importance (learnable weights and biases) to various aspects in the input image. The ConvNet consists of a set of learnable filters (also known as kernels). During the forward pass, each filter is convolved across the width and height of the input volume. It computes the dot product between the entries of the filter and the input and produces a 2-dimensional activation map of that filter. The architecture is able to capture the Spatial and Temporal dependencies in an image by applying relevant filters.

Batch normalization (BN) is a technique that improves the speed, performance, and stability of artificial neural networks by adjusting and scaling the activations Batch normalization solves internal covariate shift. During the training phase, small changes in shallow hidden layers can be amplified as we propagate within the network. This results in significant shift in deeper hidden layers. Therefore, batch normalization is proposed in order to reduce these shifts to speed up training and to produce more reliable models. Furthermore, batch normalization regularizes the network for a generalized performance and robust learning rates.

The pooling layer reduces the dimensions of the data as it combines the outputs of neuron clusters at one layer into a single neuron in the next layer. Max Pooling computes the maximum value from each of a cluster of neurons at the prior layer while Average Pooling uses the average value from each of a cluster of neurons at the prior layer.

The fully connected layer connects all the neurons in one layer to every neuron in another layer.

ReLU Activation Function is used in both models (ResNet50 and ResNet152).
The activation function of a node, in neural networks, defines the output of that node given one or more input. The generated output is then used as input for the next node. It maps the resulting values into the desired range, between -1 and 1 for example.
ReLU is the abbreviation of rectified linear unit, which is an activation function that removes negative values from an activation map by setting them to zero. ReLu eases the use of stochastic gradient descent with backpropagation of errors to train deep neural networks to allow complex the learning of relationships in the data.
ReLU(x)=max(x,0)
“ Because rectified linear units are nearly linear, they preserve many of the properties that make linear models easy to optimize with gradient-based methods. They also preserve many of the properties that make linear models generalize well.“ [16]
When applying ReLu, computations are cheap, sparse representation exists (the capability of outputting a true zero value) and the linear behavior is applied.

3.4.2 VGG16

VGG16 [20] is the third model used in the feature extraction process. It is a convolutional neural network architecture. The main idea behind this model is that it replaces large kernel-sized filters with multiple 33 kernel-sized filters one after another.

The input to cov1 layer is of fixed size 224 x 224 RGB image as shown in figure 3.11 . The image then is undergoes a stack of convolutional layers with filters having a very small receptive field: 33 (which is the minimal size to capture the notion of left/right, up/down, center). VGGs spatial pooling is done by five max-pooling layers (Max pooling calculates the maximum value from each of a set of neurons at the prior layer.). These layers are applied over a 22 pixel window with stride equaling to two.
Three Fully-Connected (FC) layers are used after the convolutional layers: the first two layers contain 4096 channels each, the third performs 1000-way ILSVRC (ImageNet Large Scale Visual Recognition Challenge) classification and thus contains 1000 channels (one channel per class since VGG is trained on the ImageNet dataset). The softmax layer is the last layer of this architecture. Softmax is a function that takes a vector of K real numbers as input, and normalizes it into a probability distribution that consists of K probabilities.
All hidden layers contain the rectification (ReLU) non-linearity. Only one layer is equipped with Local Response Normalisation (LRN) because it increases the memory consumption and the computation time [44].

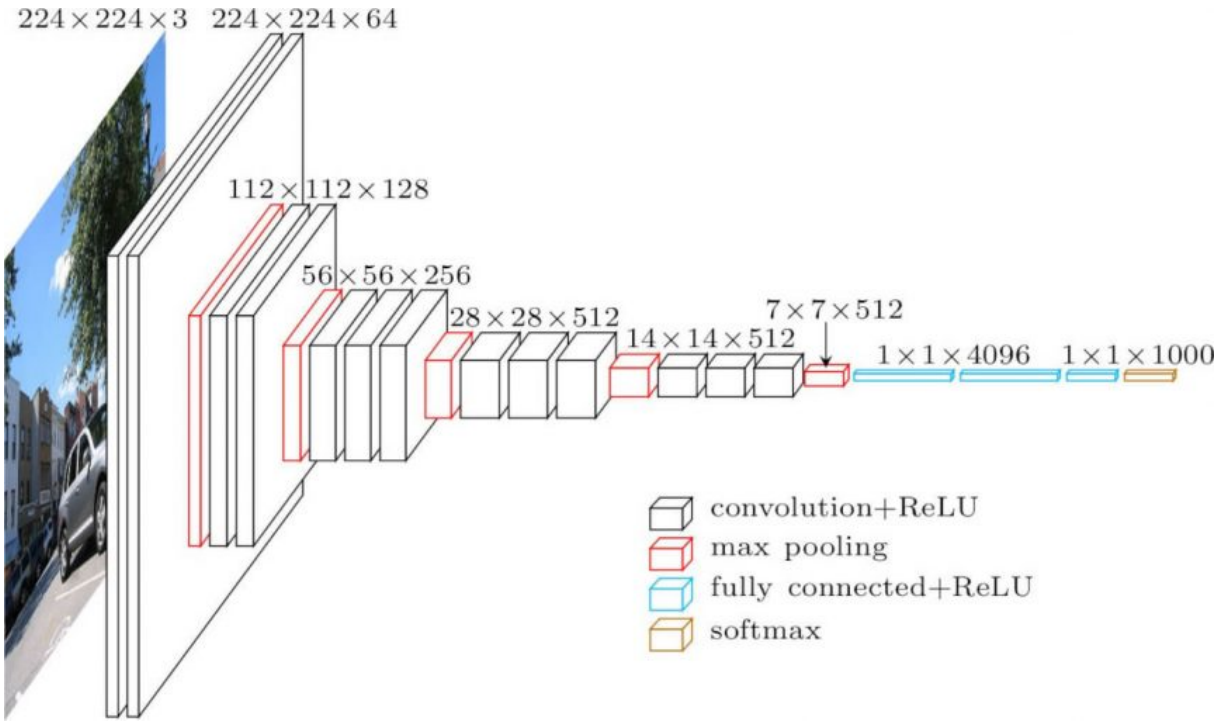


Figure 3.11: VGG16 architecture

3.4.3 Inception-BN

By creating this model, Google introduced Batch Normalization (BN) [38] to deep convolutional neural networks.
The main aim of the Inception architecture is to improve the utilization of the computing

resources in the network. This was obtained by increasing the depth and width of the network based on the Hebbian principle [29] and the concept of multi-scale processing.
The most common way of improving the performance of deep neural networks is to increase its size. However, many drawbacks exist; the network becomes more prone to overfitting and the use of computational resources dramatically increase.
The Inception model solve these issues by moving from fully connected to sparsely connected architectures as proved by Arora et al. [35] .

The network gets wider instead of deeper. The naive inception module, as shown in figure 3.12 . , performs convolution on an input, with 3 different sizes of filters (1x1, 3x3, 5x5). In addition, max pooling is also performed. The outputs are concatenated before being passed to the next inception module.

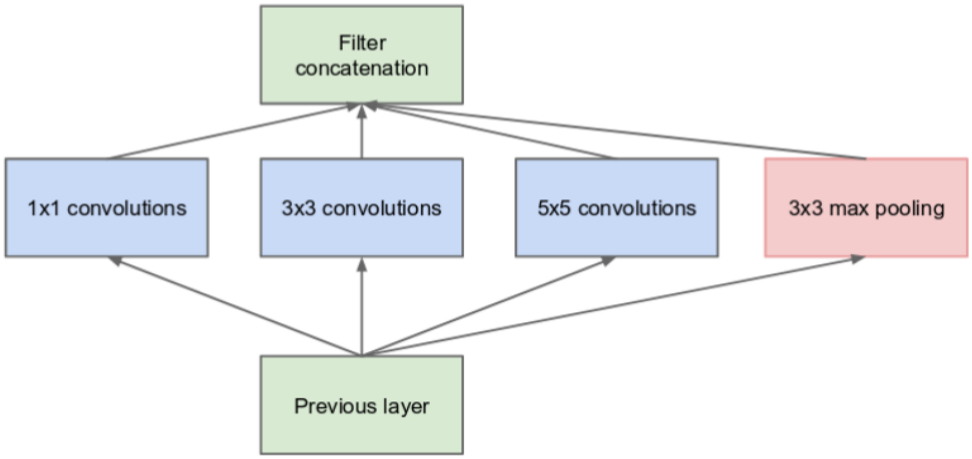


Figure 3.12: Inception module, naive version[38]

In order to make the process less computationally expensive, an extra 1x1 convolution is added before the 3x3 and 5x5 convolutions, as shown in figure 3.13 . Though adding more operations may seem counter-intuitive, 1x1 convolutions are cheaper than 5x5 convolutions, and the number of input channels is therefore reduced.

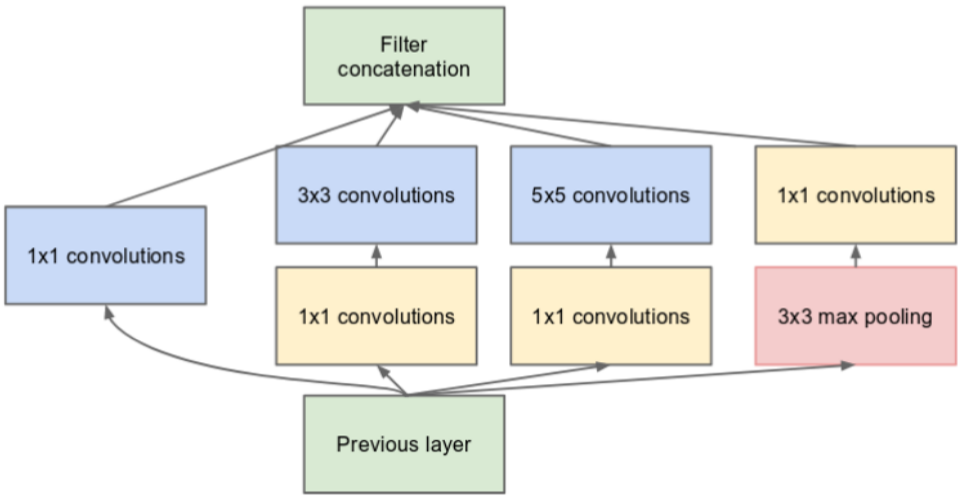


Figure 3.13: Inception module with dimension reductions[38]

Additionally, Batch Normalization (BN) layers are added in the Auxillary Classifiers. BN is used for normalizing the value distribution before passing through the next layer. Therefore, higher accuracy and faster training speed can be achieved. By applying BN, the input X is multiplied by weight W and added by bias b and become the output Y at the next layer after an activation function F:
Y=F(W X+b)
F is a sigmoid activation function which can be saturated at 1 making the gradient become zero. As we increase the depth, the vanishing gradient problem appears. To avoid this problem, ReLU is then used as F.

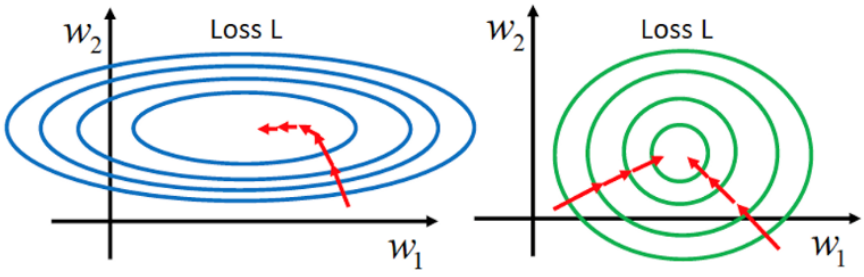


Figure 3.14: Without BN (Left), With BN (Right) [38]

X must remain fixed over time for a negligible change to be amplified when the network becomes deeper. Therefore, a higher learning rate can be achieved and the need for Dropout is reduced.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure 3.15: Batch Normalization [38]

During training (as shown in figure 3.15), the mean and variance of the mini-batch are estimated. The input is then normalized by subtracting the mean and dividing by the standard deviation . (The epsilon is to prevent denominator from being zero) And additional learnable parameters y and B are used for scale and shift in order to achieve a better shape and position. And output Y becomes as follows:

$$Y = F(BN(W * X + b))$$

3.4.4 Feature Extraction Implementation

First of all, an instance of the model, either ResNet50, ResNet152, VGG16 or Inception-BN, is created and the weights are set to the ImageNet database parameters. The loaded model contains the default symbol which returns the output of the last layer. Since we are using the model to extract features, we are interested in getting the internal outputs from the neural network rather than the final predicted probabilities. Therefore, we can get all the internal layers and create a new module with a new symbol. In our case, the output before the last fully connected layers (Flatten-0 for the stated models) is used as it returns semantic features of the raw images which are not fit to the ImageNet labels yet. Hence, we can obtain the feature vector (a list of 2048 floating point values) rather than the classification class.

3.5 Feature Reduction

Feature selection helps in identifying the related features from a set of data and removing the irrelevant features. It is a core concept in machine learning which has a direct

impact on the models performance. Irrelevant or partially relevant features can negatively influence the accuracy of the model.
Feature selection is performed for multiple reasons. First of all, it reduces overfitting as it reduces the redundant data and therefore the decision is not based on noise. In addition, less misleading data achieve a higher accuracy. It also reduces the models complexity and, hence, the training time is decreased.
In this thesis, several feature selection techniques were exploited to get the best results. These techniques include SelectKBest and PCA.

3.5.1 SelectKBest

Select features according to the k highest scores of a defined function. The score function used is Chi-square (chi2).
Chi-square is calculated between each feature and the target and select the desired number of features with best Chi-square scores. In statistics, Chi-square is used to determine if the association between two categorical variables of the sample would reflect their real association in the population.
Chi- square score is given by:

$$X^2 = \frac{(ObservedFrequency - ExpectedFrequency)^2}{ExpectedFrequency}$$

where
Observed frequency = No. of observations of class.
Expected frequency = No. of expected observations of class.

3.5.2 Principal Component Analysis (PCA)

PCA [17] is a basic technique in exploratory data analysis (EDA) that reduces the feature space and tries to find the directions of the variation in the dataset. Thus, dummification (the vast amount of irrelevant data produced by the feature extraction process) is decreased.
In statistics, Principal component analysis (PCA) is a procedure that uses an orthogonal transformation in order to convert a set of dependent features into a set of linearly uncorrelated features (known as principal components) in a new coordinate system.
Additionally, these new dimensions align with the direction of the maximal variation and this is the main benefit behind PCA. Hence, given a set of features (2048 in our case), only the most important dimensions are kept.
The first principal component has the largest variance, and each succeeding component has the highest possible variance that is orthogonal to the previous components as an analogue to the eigenvalue decomposition in linear algebra.
In this thesis, PCA was computed using the covariance method. We first organize the

dataset in n data vectors from X_1 to X_n (where n is the number of labeled mammograms) as row vectors, each of which has p variables (where p represents the number of features learned from the transfer learning phase). Now, we have a matrix of n x p dimensions. Afterwards, the empirical mean is calculated along each column j according to the following formula:

$$u_j = \frac{1}{n}\sum_{i=1}^{n} X_{ij}$$

We then calculate the deviations from the mean by centering the data by subtracting the empirical mean vector U from each row of the data matrix X.

$B = X - hu^T$
where h is n x 1 column vector of all 1s:

$h_i = 1$ for i - 1,....,n

The next step is to calculate the covariance matrix C from the matrix B according to the formula:

$$C = \frac{1}{n-1} B * B$$

Where * is the conjugate transpose operator (the regular transpose in this particular case as we have real numbers only). Also, $n-1$ is used instead of n according to Bessel's correction (a bias correction approach).

The matrix V of eigenvectors is then found which diagonalizes the covariance matrix C:

$$V^{-1}CV = D$$

where D is the diagonal matrix of eigenvalues of C. Finally, we compute the cumulative energy content for each eigenvector. The cumulative energy content g for the the eigenvector is the sum of the energy content across all of the eigenvalues from 1 through j:

$$g_j = \sum_{k=1}^{j} D_{kk} for j = 1, ..., p$$

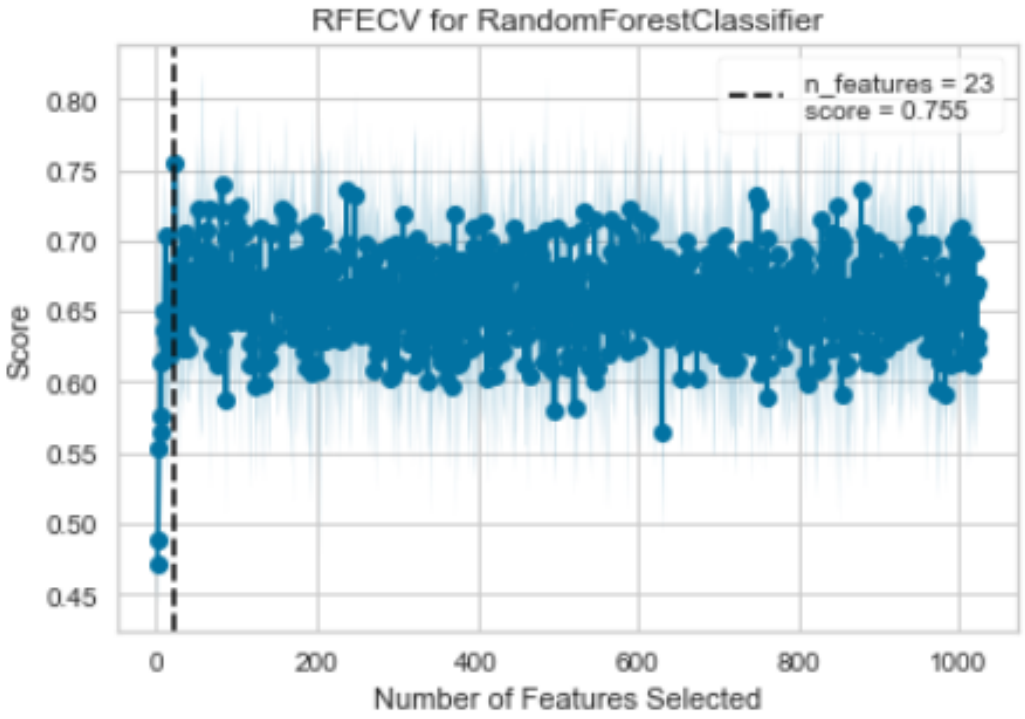
A subset of the eigenvectors is selected as basis vectors and the Z-scores of the data are projected onto the new basis. We choose the smallest value L that lies between 1 and p by the aid of the vector g, where the cumulative energy g is above a certain threshold (90 percent) as shown below .These last steps were computed by a pre-implemented methods in sklearn library.

$$\frac{g_L}{g_p} > 0.9$$

To conclude, PCA is an eigenvector problem. It explains the variance in the data by revealing the internal structure of the data. When the dataset has a set of coordinates in a high-dimensional data, PCA supplies a lower-dimensional description. This approach is implemented by using only the first few principal components so that the dimensionality of the transformed data is decreased.

3.5.3 Number of Features by Cross-Validation

Feature extractors require a specified number of features to keep, however it is not known in advance how many features are valid. To find the optimal number of features cross-validation is used to score different feature subsets and select the best scoring collection of features as shown in the graph below. Our main aim is to select features by recursively considering smaller and smaller sets of features. First of all , the estimator is trained on the complete set of features (2048 features) and the importance of each feature is obtained through a feature importances attribute. Then, the least important features are pruned from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.



3.6 Classification

Classification is the process of predicting the class (also known as target or labels) of a given input. For mammogram images, the classification is performed on two phases. The first phase predict whether the mammogram is normal or abnormal; if the result is abnormal, a second classification phase decides whether it is benign or malignant.

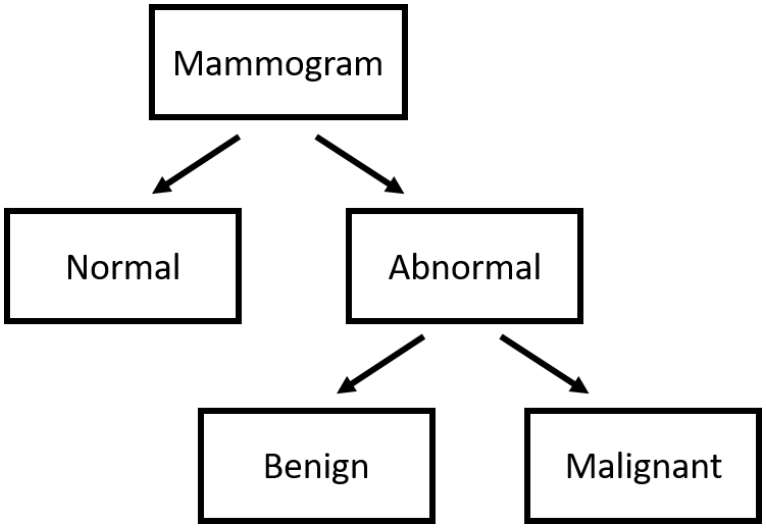


Figure 3.16: Classification Diagram

Classification is a part of supervised machine learning where the labels must be provided with the input data. This means that when we implement the classifier, each feature vector must be accompanied with its label.
There exist mainly two types of classifiers: lazy learners and eager learners.
Lazy learners are the basic classifiers. They store the training data until a testing data is provided. Then, classification is performed based on the most related data in the training data. This approach was tested with K-nearest neighbor and was then excluded from the trials since it takes more time in the prediction process than the training phase which is not suitable for a user application.
Therefore, eager learners were implemented. By using this approach, a model based on the training data is constructed before receiving the test data. Only one hypothesis is made and covers the entire instance space. Thus, a long time is taken in training but a relatively small time is needed for prediction.
The main classifier used in this thesis is Random Forest. Nonetheless, Naive Bayes,Logistic Regression, Support-Vector Machines and K-nearest neighbor were also implemented with the intention of optimizing the classification accuracy.

3.6.1 Random Forest (RF)

Random forests or random decision forests are an ensemble learning method used for classification and regression tasks. At training time, the RF algorithm constructs a set of

decision trees. The mode of the classes (for classification) and the mean prediction (for regression) of each tree are generated as the output.
This algorithm was first developed by Tin Kam Ho [15] by implementing the stochastic discrimination [21] approach to classification. Ho corrected the decision trees habit of overfitting the training data by establishing that forests of decision trees splitting with oblique hyperplanes are capable of gaining a high accuracy, under the constraint that the forests are randomly restricted to be sensitive to only selected feature dimensions. Therefore, the common belief about classifiers being susceptible to overfitting when the complexity grow (larger forest) is disapproved. This method can be explained by Kleinbergs theory of stochastic discrimination [21].

RF Preliminaries:

The Stochastic Discrimination [21], in pattern recognition, is a methodology used to construct classifiers. The base idea is to randomly combine arbitrary numbers of weak components, to generalize to new data while retaining the characteristics of the weak components via the Central Limit Theorem [7].

Decision trees are a predictive modeling approach. In machine learning and data mining, decision trees take a set of discrete values as the classification target from several input variables; class labels are represented in the leaves while the branches contain the conjunctions of features. Each internal node is labeled with an input feature. Each arc leaving the node is labeled with each of the possible target values. The learning is done by splitting the source training set into small subsets. The splitting is then repeated on each subset recursively; this process is known as recursive partitioning. The recursion stops when the base case is reached, either when the subset at a node reaches the same target value or when partitioning no longer adds value to the prediction.
Decision trees approach is an application of greedy algorithm due to its top-down induction of decision trees (TDIDT). It uses the basic "divide and conquer" top-down recursive pattern:
1. Take all the dataset as input.
2. Find a split based on an attribute that maximizes some purity measure.
3. Take the output to the input data (the "divide" step).
4. Re-apply steps 1 and 2 to each split (the recursive "conquer" step).
5. Then prune back to reduce overfitting.

Bootstrap aggregating (bagging) technique is applied to the training algorithm for random forests. It is a technique designed to enhance the accuracy and the stability of machine learning algorithms.
Given a training set X = x_1, ..., x_n along with responses Y = y1, ..., yn, bagged B times (For b = 1, ..., B), the algorithm chooses a random sample with replacement of the training data and fits trees to these samples. X_b and Y_b are the samples of n training examples from X and Y. Then, a classification tree F_b is trained on X_b and Y_b. Now, a prediction can be made by the majority vote (mode).
Variance and overfitting are also reduced without increasing the bias; while the predictions made by one tree are highly sensitive to noise, the average of many trees is not. Bootstrap

sampling de-correlates the trees by passing different training subsets.
The number of trees (B) typically ranges from a few hundred to several thousand trees. The optimal number of trees B was found in this thesis using cross-validation (details are found below).

Random forests differ from the bagging method by selecting a random subset of the features at each candidate split. This process is known as feature bagging.
The random subspace method (feature bagging) is an ensemble learning method in machine learning that reduces the dependency between estimators in an ensemble. Estimators are trained on random sets of features instead of the whole feature set to avoid the following problem. If some features are considered as strong predictors for the target variable, these features will be chosen in many of the B trees, causing a high correlation.

RF Properties:

When fitting a random forest to the data, the **out-of-bag error** (the mean prediction error on each training sample x, using only the trees that did not have x in their bootstrap sample) for each data point is computed, recorded and averaged over the forest.

In order to compute the **importance of a feature** after training, the values of this feature are permuted among the training set and out-of-bag error is re-computed. The feature importance is calculated by averaging the difference in out-of-bag error before and after the permutation over all trees. The score is then normalized by the standard deviation of these differences. More important features produce large score values. Sklearns library measures the features importance by computing the previously-mentioned score then scales the results for the sum to be equal to 1.

RF Hyperparameters:

The hyperparameters in random forest are used for two main reasons: increasing the predictive power of the model and making the model faster. The hyperparameters tackled in this thesis are the ones associated with the sklearns built-in random forest function.

In order to increase the predictive power, three hyperparameters are taken into consideration.
1. The **n-estimators** hyperparameter represents the number of trees built before voting. Generally, the large number of trees increases the performance and stability of predictions. However it increases the computational time. The n-estimators values range from 200 to 2000 depending on the cross validation output.
2. The **max-features** hyperparameter is the maximum number of features used when splitting a node. The max-features hyperparameter used is 'sqrt' upon the cross validation output.
3. The **min-sample-leaf** hyperparameter determines the minimum number of leafs (10 in this project) that are required to split an internal node.

In order to increase the models speed, three other hyperparameters were calculated.
1. The **n-jobs** hyperparameter represent the number of processors used. A value of 1

means that the engine can only use one processor. A value of -1 was used meaning that there is no limit for the number of processors.
2. The **random-state** hyperparameter makes the models output replicable. The same results are produced at every running time.
3. The **oob-score** is a random forest cross validation method. It computes the out-of-bag score. It was set to TRUE in order to be activated.

Tuning RF Hyperparameters:

The first steps execued in order to improve the performance were gathering more data and feature engineering. However, when all data sources are exhausted, the hyperparameters must be set before the training starts.
This phase is considered to be a trial-and-error based engineering process as it relies more on experimental results than theory. Hence, different combinations must be evaluated to determine the optimal settings of a Random Forest. Accordingly, the standard procedure for hyperparameter optimization accounts for overfitting through cross validation.

Cross validation (CV) was implemented in this bachelor project using K-Fold CV. In **K-Fold CV**, the training set is split into K number of subsets, known as folds. The model is fitted K times iteratively, each time training the data on K-1 of the folds and evaluating on the Kth fold (known as the validation data), as shown in figure 3.17.
The performance is finally averaged on each of the folds to get the final validation metrics for the model.

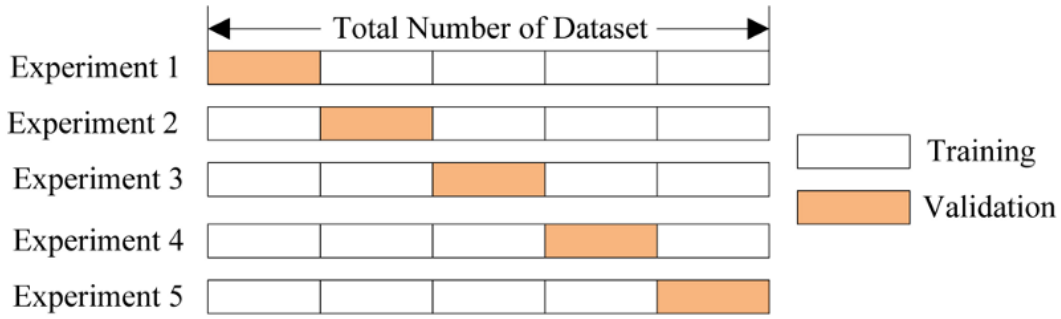


Figure 3.17: 5 Fold Cross Validation [36]

Using Scikit-Learns RandomizedSearchCV method, a grid of hyperparameter ranges can be defined to sample from during fitting. On each iteration, the algorithm selects a different combination of the features.
The **n-iter** argument in RandomizedSearchCV controls the number of different combinations; more iterations will cover a search space.
Additionally, the **cv** argument represents the number of folds used; more folds will reduce overfitting. Nevertheless, increasing both arguments augment the run time.

For additional validation, other classifiers were tested to check if Random Forest yields the best performance for mammogram classification.

3.6.2 K-Nearest Neighbors Classifier

K-Nearest Neighbors (KNN) is basic algorithm that classifies the test cases based on a similarity measure (e.g., distance functions). A case is classified by a majority vote of its neighbors. The distance function used in this thesis is the Minkowski function of order p between two variabes X and Y:

$$(\sum_{i=1}^{k}(x_i - y_i)^q)^{\frac{1}{q}}$$

KNN is a lazy classifier as stated previously. This approach was tested and was then excluded from the trials since it takes more time in the prediction process than the training phase which is not suitable for a user application. In addition, other classifiers achieved a better performance.

3.6.3 Gaussian Naive Bayes Classifier

Bayes Theorem is stated as:

$$P(h|d) = \frac{P(d|h) * P(h)}{P(d)}$$

Where
$P(h|d)$ is the probability of hypothesis h given the data d. This is called the posterior probability.
$P(d|h)$ is the probability of data d given that the hypothesis h was true.
P(h) is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
P(d) is the probability of the data (regardless of the hypothesis).

The hypothesis with the highest probability is selected after calculating the posterior probability for several hypotheses. This is the maximum probable hypothesis and may formally be called the maximum a posteriori (MAP) hypothesis:

$$MAP(h) = max(P(h|d))$$

Naive Bayes is a classification algorithm for binary problems. It is called naive Bayes (or idiot Bayes) because the calculation of the probabilities for each hypothesis are simplified to ease the calculations.

3.6.4 SVM Classifier

Support-vector machines (SVMs) [42] are supervised learning models that analyze data used for classification and regression tasks. During the training stage, SVM creates a

model that assigns new examples to the target categories. Hence, it is considered to be a non-probabilistic binary linear classifier. The main objective of SVM is to find a hyperplane in an N-dimensional space (where N is the number of features) that distinctly classifies the data points by selecting the hyperplane (with N-1 dimensions) with the maximum margin as shown in the following figure.

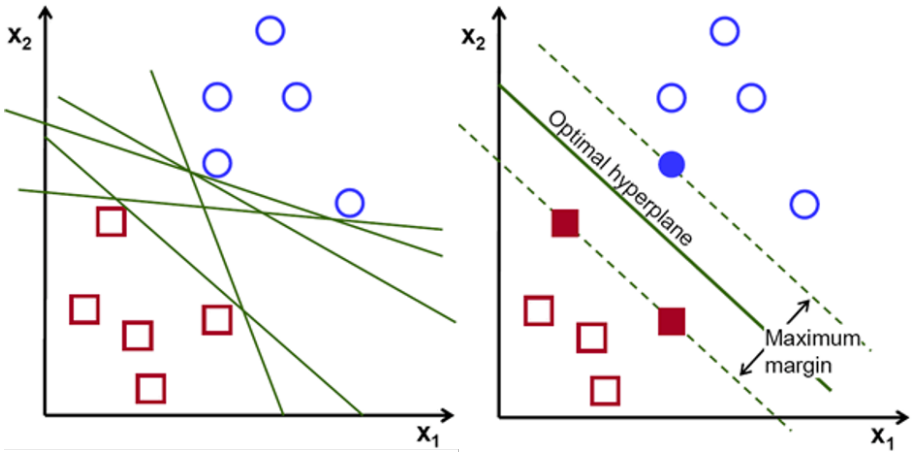


Figure 3.18: Possible hyperplanes [13]

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. These vectors help in maximizing the margin of the classifier.

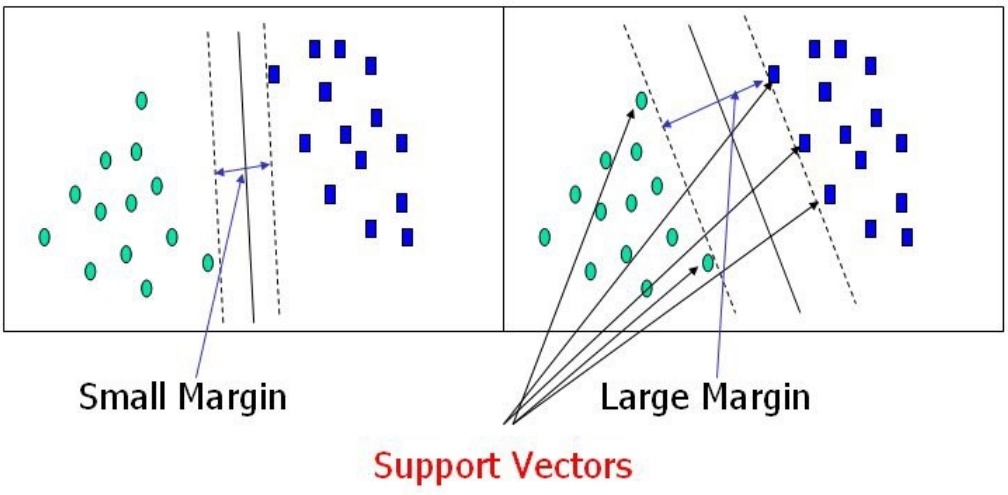


Figure 3.19: support vectors [13]

In SVM, if the output of the linear function is greater than 1, it is identified with one class and if the output is -1, it is identified with the other class. Since the threshold

values are changed to 1 and -1, the reinforcement range of values ([-1,1]) acts as margin. In order to maximize this margin between the data points and the hyperplane, the hinge loss function is used:

$$c(x, y, f(x)) = (1 - y * f(x))_+$$

A regularization parameter lambda is added to the cost function to balance the margin maximization and loss. The cost function becomes:

$$min_w \lambda ||w||^2 + \sum_{i=1}^{n} (1 - y_i(x_i, w))_+$$

Finally, partial derivatives are taken with respect to the weights to find the gradients. We can then update our weights using the gradients.

$$\frac{\delta}{\delta w_k} \lambda \parallel w \parallel^2 = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k} \left(1 - y_i \langle x_i, w \rangle\right)_+ = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

3.6.5 Logistic Regression Classifier

In Supervised Machine Learning, Logistic Regression (LR) is a Statistical Learning technique used for classification tasks. A contradiction exists between the LR name which contains the term Regression and its usage for classification. Simply, LR uses a linear regression function in order to produce discrete binary outputs.

Inputs x are continuous feature-vectors of length K, where j=1,,k and i=1,,n. The input matrix is X contains N number of data points with K number of features each. Inputs can be illustrated as:

$$X = \begin{bmatrix} x_{11} & x_{12} & . & . & . & x_{1k} \\ x_{21} & . & & & & . \\ . & & . & & & . \\ . & & & . & & . \\ . & & & & . & . \\ x_{n1} & . & . & . & . & x_{nk} \end{bmatrix}_{n \times k}, Y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix}_{n \times 1}$$

And the output y is a discrete binary variable, such that y 0,1.

The function drawn in figure 3.20 is the logistic function (also known as the sigmoid function).

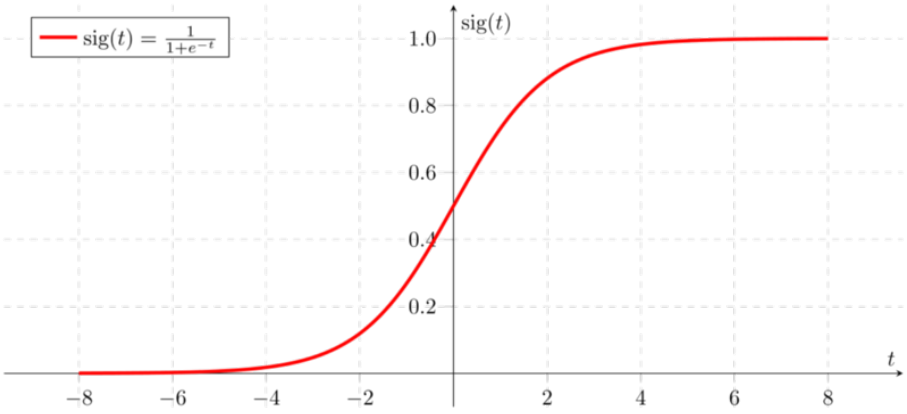


Figure 3.20: Logistic Sigmoid Activation Function [41]

The main objective is to shrink real valued continuous inputs into a range of (0,1) to transform the task into a probabilistic problem. Posteriors can then be written as:

$$P(Y|X) = \frac{1}{1 + e^{-f(x)}}$$

where f(x) is a function consisting the features Xj and their corresponding weights β in a linear form:

$$f(x) = X_0 + x_1\beta_1 + ... + x_k\beta_k + \epsilon$$

Note that x, β, f(x) and ϵ represent the random error process (noise).
By using the posterior equation, the estimation function f(x) can become a posterior probability function (known as the log-of-odds ratio):

$$log[\frac{P(Y|X)}{1 - P(Y|X)}] = x_0 + X_1\beta_1 + ... + x_k\beta_k + \epsilon = f(x)$$

Logarithmic transformation is used for three main reasons. It implements normalization (scaling) making the coefficients more consistent. Also, operations become easier to perform, multiplications become summations, divisions become subtractions and most importantly exponents become multiplications. Furthermore, the resulted hyperplanes are monotone (they increase or decrease monotonically) and can be considered as a representative of the original function.

Chapter 4

Results

4.1 Experiment Setting and Dataset

4.1.1 Tools

Python was used to develop the algorithm, as it offers a solid number of machine learning and image processing libraries and packages.

4.1.2 Dataset

The MIAS dataset was used as stated in the methodology section. In the first classification stage (Normal vs Abnormal), 504 Regions of interest (ROIs) were used; 413 are normal and 91 are abnormal. In the second classification stage (Benign vs Malignant), 91 ROIs were available; 40 are malignant and 51 are benign.

4.1.3 Limitations

A limitation occurred due to the small size of the dataset (91 ROIs) in the second classification phase. Hence, more abnormal ROIs were added from the InBreast dataset. The final data contains 141 ROIs; 86 are malignant and 55 are benign.
These numbers are still relatively small and negatively affect the second classification performance as it is known that large amount of training data plays a critical role in making the Deep learning models successful. This limitation gives the models high variance, less power and represents underfitting.
We tried to overcome this limitation by applying transfer learning, ensembling, balancing the class weights and applying efficient loss functions (as explained in chaper 3). However, gathering more data remain a basic factor for a better performance.

4.2 Experiments and Results

Several models in each phase were tested to achieve a high performance. All different combinations were made between 2 classification tasks, 4 feature extraction models, 3 feature reduction algorithms and 6 classifiers. This resulted in 144 combinations as shown in the following figure.

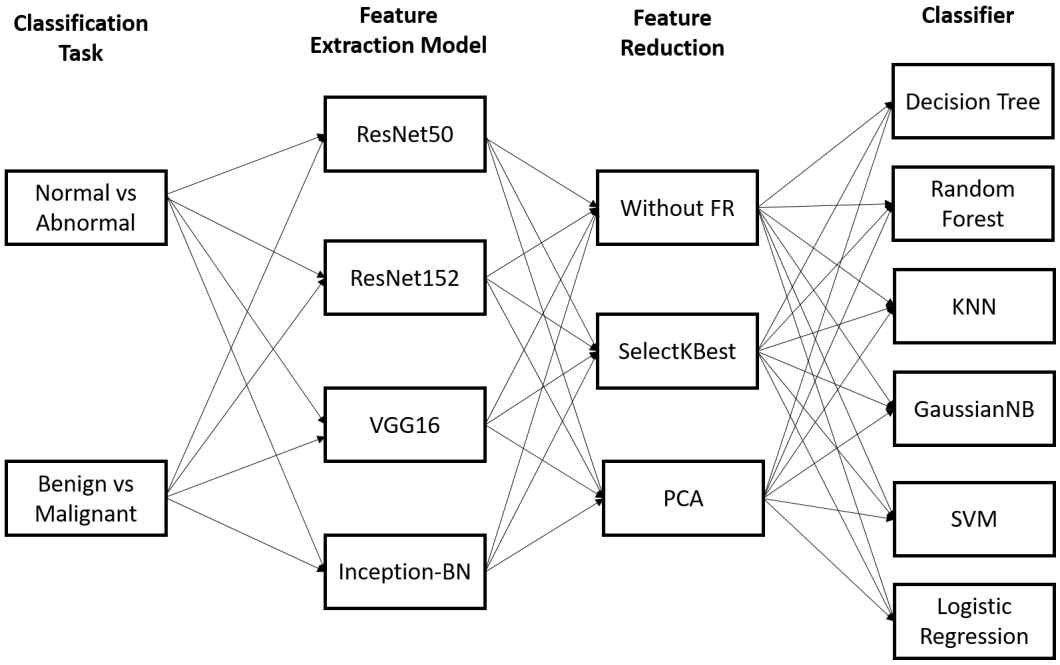


Figure 4.1: Combinations

The following tables contain the scores evaluated by cross-validation (explained in chapter 3) of the 144 combinations.
The cross-validation outputs the AUC (Area Under the Receiver Operating Characteristics) scores. AUC - ROC curve is a performance measurement for classification tasks. ROC is a probability curve and AUC represents degree or measure of separability. It measures the model capability of distinguishing between classes. Higher the AUC scores denote a better performance (the model is able to distinguish between normal, benign or malignant mammograms). The ROC curve is plotted with True Positive Rate (y-axis) against the False Positive Rate (x-axis).

Table 4.1: cross-validation scores without feature reduction (first classification)

Normal vs Abnormal	ResNet50	ResNet152	VGG16	Inception-BN
Decision Tree	93.72	97.25	92.7	90.55
Random Forest	97.38	96.73	96.85	95.27
KNN	95.83	95.84	95.28	96.45
GaussianNB	94.28	94.29	84.84	83.04
SVM	97.03	97.05	96.84	97.83
Logistic regression	97.43	97.45	97.23	97.45

Table 4.2: cross-validation scores without feature reduction (second classification)

Benign vs Malignant	ResNet50	ResNet152	VGG16	Inception-BN
Decision Tree	55.1	49.22	60.88	47.44
Random Forest	69.13	59.5	59.66	56.25
KNN	56.89	45.78	52.56	59.0
GaussianNB	63.75	60.22	55.67	48.44
SVM	65.2	58.22	49.67	67.66
Logistic regression	67.61	59.22	51.78	62.66

Table 4.3: cross-validation scores with SelectKBest (first classification)

Normal vs Abnormal	ResNet50	ResNet152	VGG16	Inception-BN
Decision Tree	94.7	96.27	94.47	94.87
Random Forest	96.85	96.07	95.27	96.85
KNN	97.83	96.84	95.87	95.48
GaussianNB	94.89	97.04	94.9	93.14
SVM	95.87	97.05	95.68	96.85
Logistic regression	96.45	96.66	96.07	96.26

Table 4.4: cross-validation scores with SelectKBest (second classification)

Benign vs Malignant	ResNet50	ResNet152	VGG16	Inception-BN
Decision Tree	56.67	52.44	53.35	63.33
Random Forest	70.97	**71.66**	66.41	65.75
KNN	55.71	57.0	43.89	57.89
GaussianNB	63.13	68.67	63.67	56.78
SVM	66.32	63.11	58.78	69.33
Logistic regression	61.14	64.44	61.78	62.89

Table 4.5: cross-validation scores with PCA (first classification)

Normal vs Abnormal	**ResNet50**	**ResNet152**	**VGG16**	**Inception-BN**
Decision Tree	92.49	94.68	94.47	93.49
Random Forest	96.66	97.38	98.32	**97.86**
KNN	97.83	95.85	96.64	95.67
GaussianNB	96.06	94.88	95.86	93.69
SVM	97.63	95.86	96.06	94.69
Logistic regression	97.04	96.45	96.05	95.08

Table 4.6: cross-validation scores with PCA (second classification)

Benign vs Malignant	**ResNet50**	**ResNet152**	**VGG16**	**Inception-BN**
Decision Tree	52.69	52.88	67.22	47.33
Random Forest	66.27	54.0	69.41	58.75
KNN	63.09	55.0	60.89	55.78
GaussianNB	65.59	55.89	49.56	60.0
SVM	53.71	53.78	66.0	61.89
Logistic regression	59.13	54.77	64.22	58.66

For the first classification phase (normal vs abnormal), the top cross-validation scores were given by the Inception-BN feature extraction model when combined with the Random Forest classifier using the PCA feature reduction technique.The cross-validation score is 97.857.
Moreover, the top score for the second classification phase (benign vs malignant) resulted from the combination of the ResNet152 feature extractor, Random Forest classifier and SelectKBest feature reduction technique. The cross-validation score is 71.666.
Both results are marked on the previous tables.

4.3 Enhancements

Due to the small number of abnormal samples, data augmentation was applied. The available samples were rotated by different angles and flipped. This resulted in an augmented dataset of 300 mammograms where 143 are benign and 147 are malignant.
These augmentations are justified because the masses do not have any inherent orientation and their diagnosis is invariant to these kinds of transformations.
The results of the second phase were then updated as shown in the following tables.

Table 4.7: cross-validation scores without feature extraction (second classification)

Benign vs Malignant	ResNet50	ResNet152	VGG16	Inception-BN
Decision Tree	62.68	65.26	52.62	61.65
Random Forest	81.83	**83.32**	68.73	78.46
KNN	77.62	81.94	57.57	76.04
GaussianNB	67.25	65.97	60.34	56.01
SVM	77.28	78.36	64.34	74.99
Logistic regression	75.29	77.97	62.35	78.95

Table 4.8: cross-validation scores with SelectKBest (second classification)

Benign vs Malignant	ResNet50	ResNet152	VGG16	Inception-BN
Decision Tree	59.23	63.97	62.61	63.21
Random Forest	75.71	77.28	70.41	69.02
KNN	66.63	72.97	59.95	66.97
GaussianNB	61.72	65.64	64.98	64.98
SVM	60.91	67.68	66.99	66.27
Logistic regression	61.94	66.99	67.35	64.98

Table 4.9: cross-validation scores with PCA (second classification)

Benign vs Malignant	ResNet50	ResNet152	VGG16	Inception-BN
Decision Tree	63.96	69.01	61.64	65.29
Random Forest	65.22	76.78	71.07	75.55
KNN	60.27	76.6	75.61	72.32
GaussianNB	51.95	63.9	61.91	59.94
SVM	52.64	61.67	64.61	53.56
Logistic regression	50.58	59.7	63.9	53.3

After data augmentation, the top score for the second classification phase (benign vs malignant) resulted from the combination of the ResNet152 feature extractor and Random Forest classifier with a cross-validation score of 83.32.

More metrics were computed to evaluate the model. This includes train and test accuracies, confusion matrix, precision, recall and F1-score.

A confusion matrix is a table used to describe the performance of a classification model. It contains four values:

-True positive (TP): A tuple predicted to be in class, and is actually in it.
-False positive (FP): A tuple predicted to be in class, but is actually not in it.
-True negative (TN): A tuple not predicted to be in class, and is actually not in it.
-False negative (FN): A tuple not predicted to be in class, but is actually in it.

Accuracy is the amount of correct classifications made from the total amount of classifications. As a heuristic, accuracy gives an overview on the performance of the model. However, it does not give detailed information regarding its application to the problem. It may also be misleading when having a class misbalance.
The train accuracy is the accuracy of a model on the training set.
The test accuracy is the accuracy of a model on the test set.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision (also known as positive predictive value) is the amount of relevant instances among the retrieved instances, while **recall** (also called sensitivity) is the part of relevant instances that have been correctly classified over the total amount of relevant instances.
In other words, precision gives good indications when the costs of false positives are high. In our case, if the model has low precision, many patients will be told that they have cancer, thus, some misdiagnoses will be included. On the other hand, recall helps when the cost of false negatives is high. In our case, if the model has low recall, some patients with abnormal lesions will be diagnosed as normal.

$$Precision = \frac{TP}{FP + TP}$$

$$Recall = \frac{TP}{FN + TP}$$

F1 score is a metric that combines precision and recall. It indicates that the model has low false positives and low false negatives.

$$F1 = 2\frac{precision * recall}{precision + recall}$$

The first classification phase metrics:
Training accuracy = 97.9
Test accuracy = 95.27
Confusion Matrix:
$\begin{bmatrix} 96 & 1 \\ 5 & 25 \end{bmatrix}$

Table 4.10: First classification report

	Precision	**Recall**	**F1-score**
Normal	0.95	0.99	0.97
Abnormal	0.96	0.83	0.89
Avg/Total	0.95	0.95	0.95

.

The second classification phase metrics:
Training accuracy = 96.66
Test accuracy = 70
Confusion Matrix:
$\begin{bmatrix} 31 & 12 \\ 15 & 32 \end{bmatrix}$

Table 4.11: Second classification report

	Precision	**Recall**	**F1-score**
Benign	0.67	0.72	0.7
Malignant	0.73	0.68	0.7
Avg/Total	0.7	0.7	0.7

4.4 Discussion on Usage, Runtime and Contribution

In order predict a new mammogram, the radiologist uploads it to the proposed model. The mammogram is then divided into several overlapping crops that represent the regions of interest (ROIs). These crops are extracted using a sliding window according to the MIAS dataset radii statistics. Finally, the model outputs the predicted class for each ROI. ROIs are then mapped back to their original locations in the mammogram. Abnormal ROIs are indicated with a bounding box as shown figure 4.2 .

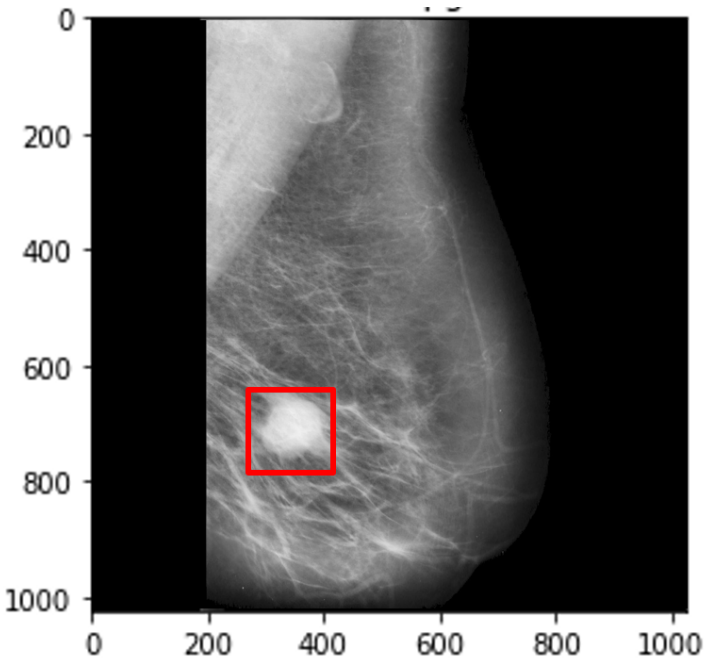


Figure 4.2: Combinations

This implementation has been tested with Tensorflow v1.10.0 on 64-bit Windows 10 Pro with Intel Core i7 CPU and 8GB RAM.
For each classification phase, the average training time is 14 minutes and 47 seconds while the average time taken to classify a new mammogram is 0.97 seconds.

In a best case scenario, when the classification output is normal, only one classification phase is performed. Hence, the algorithm takes 0.97 seconds on average to classify a new mammogram.
The worst case execution time, when a mammogram is either benign or malignant, the previous numbers are almost doubled. Two classification phases are executed in 1.62 seconds on average.
These numbers are acceptable since 1 second is the limit for the user to experience the interaction as being one continuous flow [31].

Mammography screening has been proved to reduce breast cancer risk of mortality by 38 to 48 percent among participants, according to Broeders et al [3].
However, mammograms are vulnerable to the human error caused by radiologists wrong interpretations. Consequently, in order to improve the biopsies diagnostic efficiency, double-blind reading is performed to provide a second opinion and to reduce, to some extent, the workload on the radiologist. Nevertheless, this strategy consumes a lot of resources between money, time and labor, and it remains subject to errors.

The computer-aided methodology developed in this thesis can be put into practice as an alternative for the second reader in the double-blind workflow to assist in and support the decisions made.

According to our results, the sensitivity rates of detecting abnormalities can be increased by 9 to 55 percent, depending on the level of observers (either attending, fellow or resident) according to Kundel et al [14] statistics.
Although, different methodologies and datasets cannot be directly compared, our model performed as good as the models that used the MIAS dataset along side Random Forest for detecting and classifying abnormalities. These models were stated in the literature review section. The top accuracy, in this scope, was reached by Dhungel et al [30] with 96 percent while our proposed model reached 95.27 percent.

Additionally, the median time taken by observers to hit a cancer regardless of the decision outcome is 1.13 seconds [14]. This represents the first classification phase.
Kundel also showed that the average time taken to classify the tumor is 2.37 seconds. The latter case is represented in the second classification phase. Therefore, our proposed model can surpass the human performance.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we addressed the problem of mammogram cancer detection. Two cascaded classification tasks were implemented using several deep learning techniques. A discussion on different feature extractors, feature reducers and classifiers has been provided.

The main focus of our thesis was to achieve state-of-the-art while optimizing the runtime for a user-friendly interface. This may be considered a promising aspect of computer-aided methodologies for medical images as the sensitivity rates of detecting abnormalities can be increased by 9 to 55 percent, depending on the level of observers (either attending, fellow or resident) according to Kundel et al [14] statistics.

This conclusion follows from the fact that the first classification phase (normal vs abnormal) achieved an accuracy of 95.27 percent and a cross-validation score of 97.857. These results were given by the Inception-BN feature extraction model when combined with the Random Forest classifier using the PCA feature reduction technique. The cross-validation score is 97.85 .

Furthermore, despite the limitations found in the relatively small number of abnormal samples, the top score for the second classification phase (benign vs malignant) resulted from the combination of the ResNet152 feature extractor, Random Forest classifier and SelectKBest feature reducer. The cross-validation score was 83.32 while the accuracy was 70 percent. Better results were proved to be achieved by collecting more data.

Additionally, the runtime for a mammogram classification was decreased to 1.62 seconds on average.

5.2 Future Work

Many different adaptations and experiments have been left for future work due to lack of time. Future work concerns deeper analysis of different methods and proposals. Future research should consider the potential effects of other transfer learning networks.

Also, the collection of abnormal data is certainly required to train the model in order to achieve better results. In this scope, Adversarial Augmentation can be used as stated in Jendele et al [22] work. However, this paper was newly released and the implementation was partially completed due to lack of time.

Moreover, an automation of the model can be implemented. The model can learn from the input mammograms and the different combinations of the methodologies can be updated every finite interval of time.

Appendix

Appendix A

Python Code

This section contains the implementation of the bachelor thesis, including the data exploration, the pre-processing, the features extraction, the classification and all other methods used.

April 11, 2019

0.1 1-Exploring Data

In [1]:
```
#Libraries used:

    #N-dimensional array object, linear algebra, random numbers...
    import numpy as np

    #data structures and data analysis tools
    import pandas as pd

    #interactive plots
    import matplotlib.pyplot as plt

    #handling on-disk datasets as if they were NumPy arrays
    import h5py

    #Operating system dependent functionality
    import os

    #regular expression
    import re

    #pathnames matching
    from glob import glob

    #Interactive Python for images
    from IPython.display import Image
    from IPython.display import clear_output

    #openCV for image processing
    import cv2

    #Shallow and deep copy operations
    import copy

    import time
```

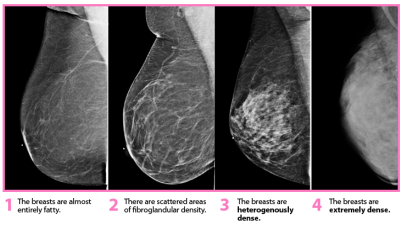
```
D:\Anaconda-Jupyter\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the seco
  from ._conv import register_converters as _register_converters
```

In [ ]:
```
#Display Dataframe
all_cases_df = pd.read_table('C:/Users/victory/Desktop/MIAS/mammogram_info.csv',
                             delimiter=' ')
all_cases_df.head(25)
```

1st column: MIAS database reference number.
2nd column: Character of background tissue:
F Fatty
G Fatty-glandular
D Dense-glandular

In [14]:
```
Image(filename='C:/Users/victory/Desktop/MIAS Help/B.PNG')
```

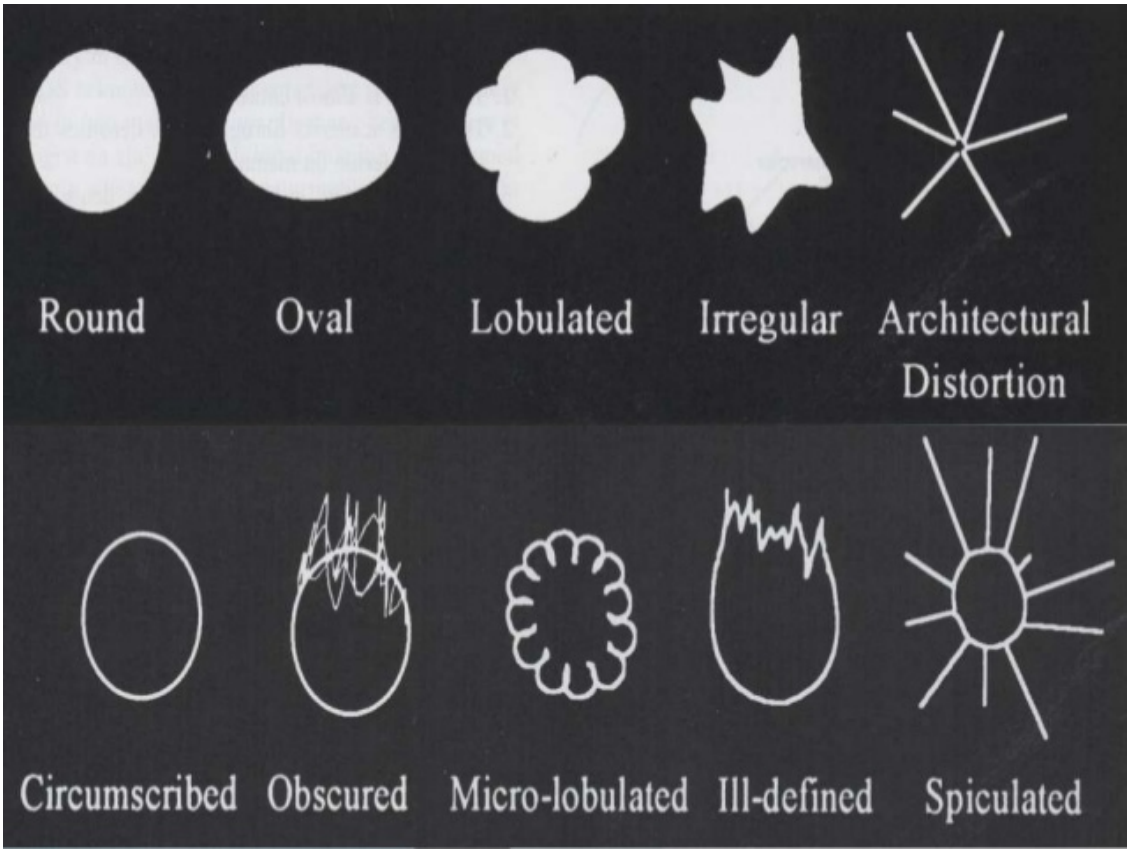
Out[14]:



3rd column: Class of abnormality present:
CALC Calcification
CIRC Well-defined/circumscribed masses
SPIC Spiculated masses
MISC Other, ill-defined masses
ARCH Architectural distortion
ASYM Asymmetry
NORM Normal

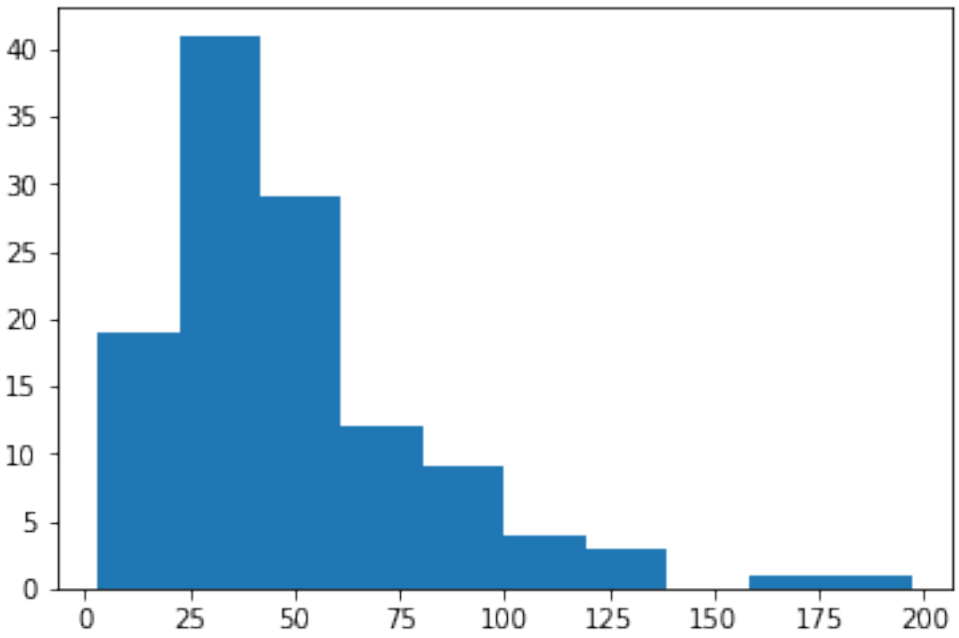
In [15]:
```
Image(filename='C:/Users/victory/Desktop/MIAS Help/B2.JPG')
```

Out[15]:

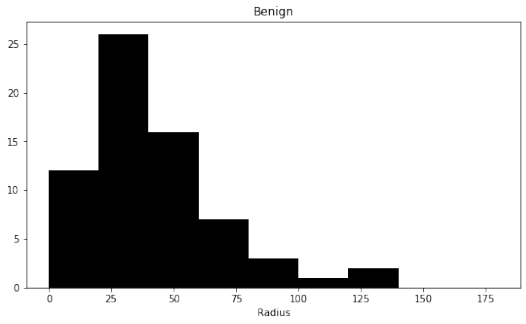


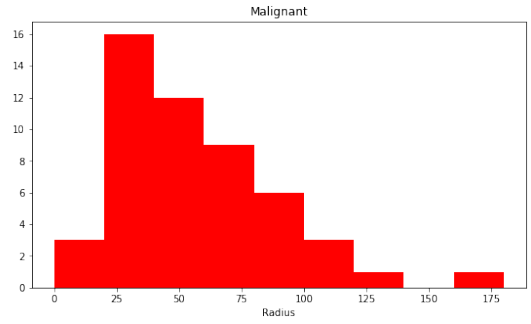
4th column: Severity of abnormality;
B Benign
M Malignant
5th, 6th columns: x,y image-coordinates of centre of abnormality.
7th column: Approximate radius (in pixels) of a circle enclosing the abnormality.

In [16]:
```
# the sizes of radii
radii = all_cases_df.Rad
radii.dropna(inplace=True)
plt.hist(radii)
plt.show()
```



In [18]:
```
#radii by severity
sev_radii = all_cases_df.loc[:,['Rad','Sever']]
sev_radii.dropna(axis=0, how='any', inplace=True)
benign = sev_radii[sev_radii.Sever == 'B']
malignant = sev_radii[sev_radii.Sever == 'M']
f, ax = plt.subplots(1,2,figsize=(20,5))
ax[0].hist(benign.Rad, color="black", bins=range(0,200,20))
ax[0].set_xlabel("Radius")
ax[0].set_title("Benign")
ax[1].hist(malignant.Rad, color="red",  bins=range(0,200,20))
ax[1].set_title("Malignant")
ax[1].set_xlabel("Radius")
plt.show()
```

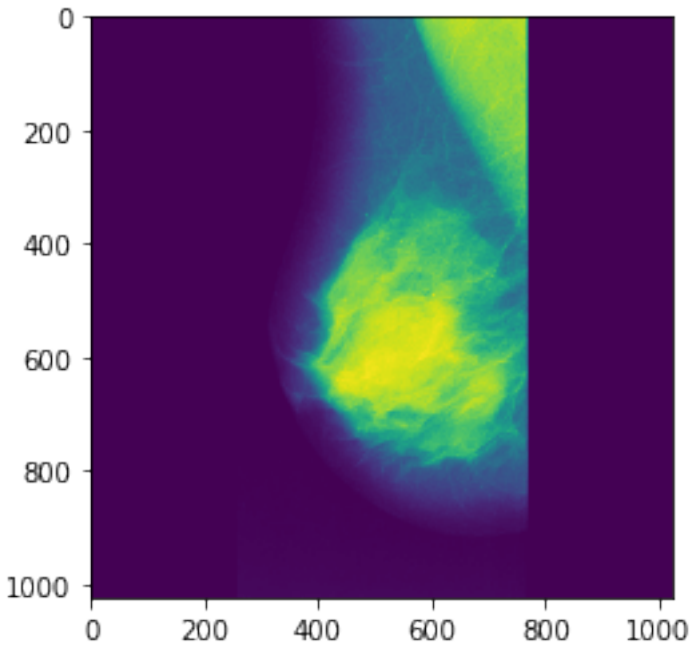




In [19]:
```
#import PGM files and return a numpy array
def read_pgm(filename, byteorder='>'):

    with open(filename, 'rb') as f:
       buffer = f.read()
    try:
        header, width, height, maxval = re.search(
            b"(^P5\s(?:\s*#.*[\r\n])*"
            b"(\d+)\s(?:\s*#.*[\r\n])*"
            b"(\d+)\s(?:\s*#.*[\r\n])*"
            b"(\d+)\s(?:\s*#.*[\r\n]\s)*)", buffer).groups()
    except AttributeError:
        raise ValueError("Not a raw PGM file: '%s'" % filename)

    image = np.frombuffer(buffer,
                          dtype='u1' if int(maxval) < 256 else byteorder+'u2',
                          count=int(width)*int(height),
                          offset=len(header)
                          ).reshape((int(height), int(width)))

    image_id = int(re.findall('([\d]+)\.', filename)[0])

    return image
```

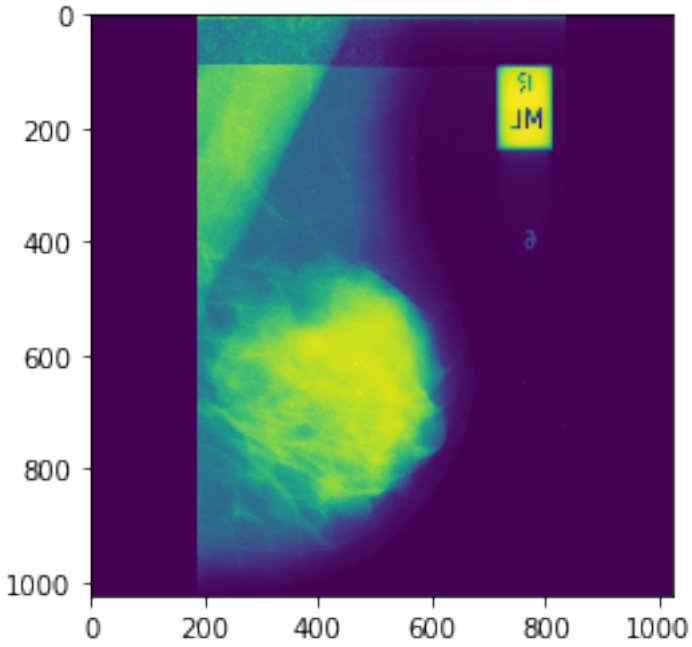
In [20]:
```
plt.imshow(read_pgm(glob('C:/Users/victory/Desktop/MIAS/*.pgm')[0]))
```

Out[20]: <matplotlib.image.AxesImage at 0x1a02d33b0b8>



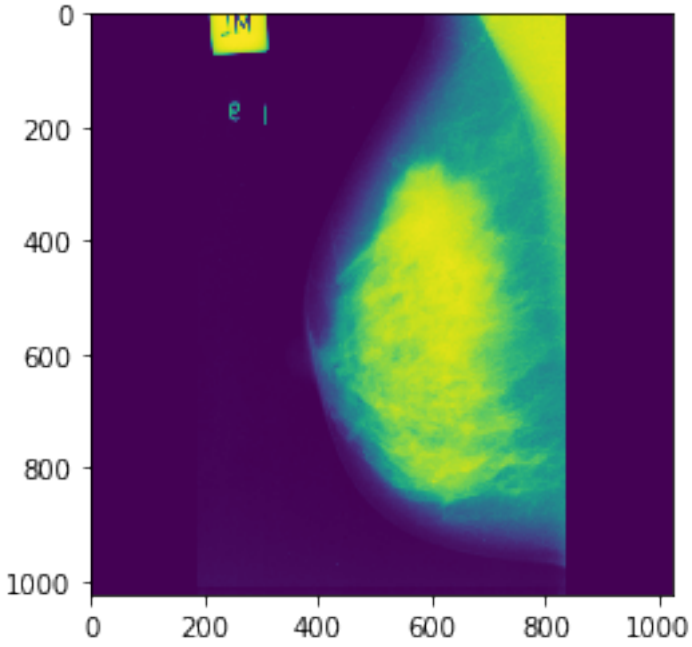
In [21]: plt.imshow(read_pgm(glob('C:/Users/victory/Desktop/MIAS/*.pgm')[1]))

Out[21]: <matplotlib.image.AxesImage at 0x1a02d2fa7f0>



In [22]: plt.imshow(read_pgm(glob('C:/Users/victory/Desktop/MIAS/*.pgm')[2]))

Out[22]: <matplotlib.image.AxesImage at 0x1a02d223240>



```
In [ ]: # Clean up the data.
        # drop images with multiple abnormalities
        # keep one row per image
        all_cases_df.drop_duplicates(subset=['Reference'],
                                     keep='first', inplace=True)

        # reindex the image
        all_cases_df.reset_index(inplace=True)
        all_cases_df.shape
```

```
In [29]: # percentages by class
         all_cases_df['Class'].value_counts(normalize=True)
```

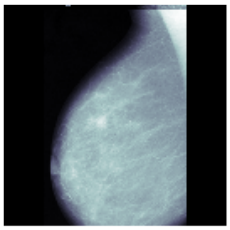
```
Out[29]: NORM    0.642857
         CALC    0.077640
         CIRC    0.071429
         SPIC    0.059006
         ARCH    0.059006
         ASYM    0.046584
         MISC    0.043478
         Name: Class, dtype: float64
```

```
In [30]: # info on the size of the abnormalities
         print("Mean abnormality radius:",np.mean(radii))
         print("Median abnormality radius:",np.median(radii))
         print("Min abnormality radius:",np.min(radii))
         print("Max abnormality radius:",np.max(radii))
```

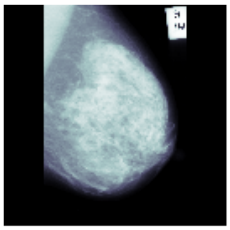
```
Mean abnormality radius: 48.84873949579832
Median abnormality radius: 41.0
Min abnormality radius: 3.0
Max abnormality radius: 197.0
```

```
In [14]: sample_count = 3
         fig, m_axs = plt.subplots(len(all_cases_df['Class'].value_counts()),
                                   3, figsize = (12, 20))
         for c_axs, (c_cat, c_df) in zip(m_axs, all_cases_df.groupby('Class')):
             for c_ax, (_, c_row) in zip(c_axs, c_df.sample(sample_count).iterrows()):
                 c_ax.imshow(read_pgm('C:/Users/victory/Desktop/MIAS/' +
                                      c_row['Reference'] + '.pgm'), cmap = 'bone')
                 c_ax.axis('off')
                 c_ax.set_title('{Class}-{Sever}'.format(**c_row))
```

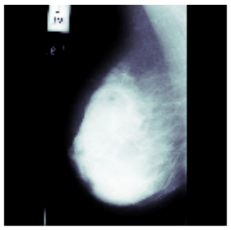
ARCH-M



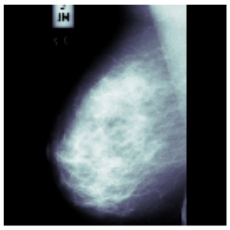
ARCH-B



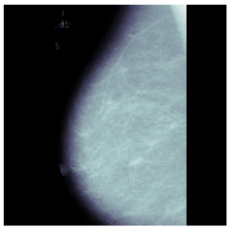
ARCH-M



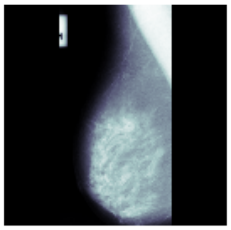
ASYM-B



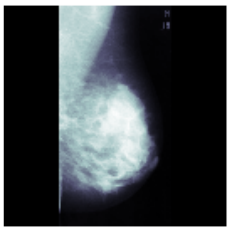
ASYM-B



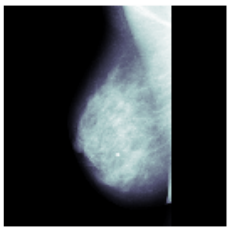
ASYM-B



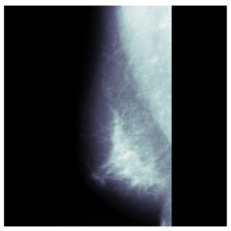
CALC-B



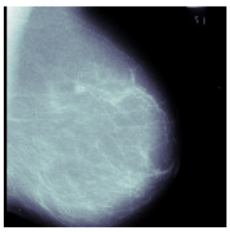
CALC-B



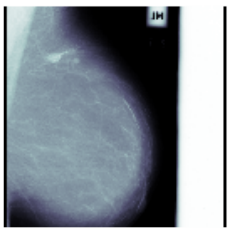
CALC-M



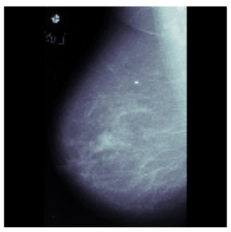
CIRC-B



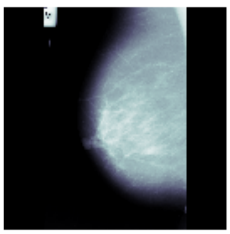
CIRC-B



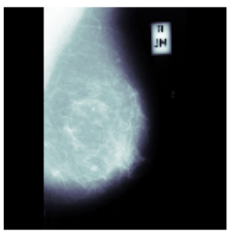
CIRC-B



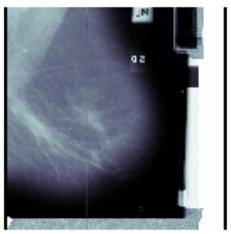
MISC-B



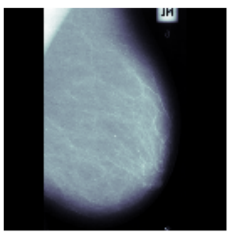
MISC-B



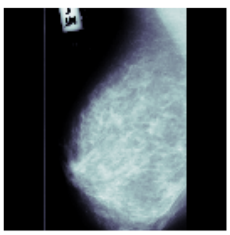
MISC-M



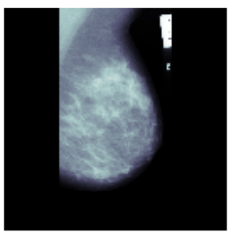
NORM-nan



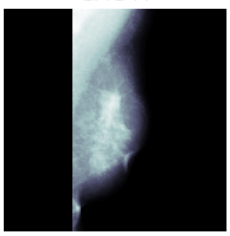
NORM-nan



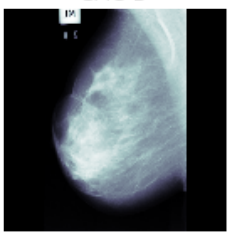
NORM-nan



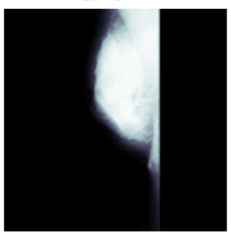
SPIC-M



SPIC-B



SPIC-M



0.2 2-PreProcessing

In [31]:
```
#create the histogramo the image
def get_hists(image, b):
    hist, bins = np.histogram(img.flatten(), bins=b, range=[0,255])
    cdf = hist.cumsum() #cumulative distribution function as ndarray
    cdf_normalized = cdf *hist.max()/ cdf.max() #normalization step

    return [hist, cdf_normalized]
```

In [32]:
```
#plot the histogram and the image
def plot(img, img_hists):
    plt.figure(1)
    plt.subplot(121)
    plt.imshow(img, cmap='gray')

    plt.subplot(122)
    plt.plot(img_hists[1], color = 'b')
    plt.plot(img_hists[0], color = 'r')
    plt.xlim([0,256])
    plt.legend(('cdf','histogram'), loc = 'upper left')

    plt.subplots_adjust(top=0.92, bottom=0.08,
                        left=0.10, right=0.95,
                        hspace=0.25, wspace=0.35)
```

In [ ]:
```
#CLAHE algorithm
def threshold(img_list, factor = 0.7, select_files = []):
    images_t = []

    def internal(data):
        thresholded = cv2.threshold(data['clahe_img'],
                                    np.median(data['clahe_img']) * factor, 255,
                                    cv2.THRESH_BINARY)[1]     # the binary image

        _, l, s, _ = cv2.connectedComponentsWithStats(thresholded)
        images_t.append( {'filename': data['filename'],
                          'clahe_img': data['clahe_img'],
                          'thresh_img': thresholded,
                          'factor': factor,
                          'labels':l,  # contiguous regions in mammogram
                          'count':s[:, -1]  # count of pixels
                         })

    if not select_files:
```

```
            print ('Processing all files')
            for i, data in enumerate(img_list):
                internal(data)

        else:
            print('Processing select files {}'.format(select_files))
            for i, data in enumerate(img_list):
                if data['filename'] in select_files:
                    internal(data)

        return images_t
```

In [34]:
```
#add the mask
def mask(image, labels, region):
    labels = copy.deepcopy(labels)  # create a full, unique copy of labels
    for row in range(image.shape[0]):
        for col in range(image.shape[1]):
            if labels[row, col] != region:
                labels[row, col] = 0  # mask the artifact
            else:
                labels[row, col] = 1  # retain the breast
    return labels
```

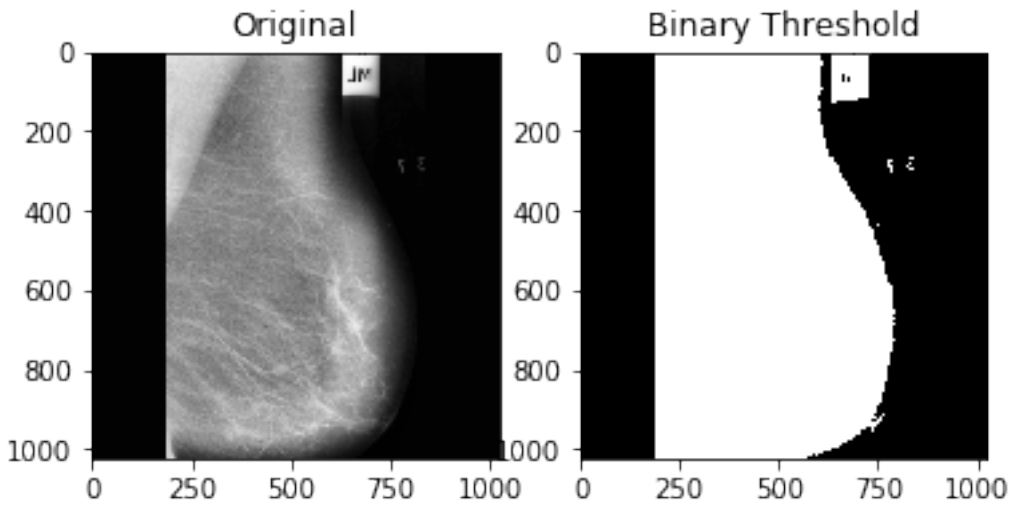
In [35]:
```
#remove all unnecesary artifacts, labels, writings...
def clean_art(images_thresh):
    revist = []
    for i, data in enumerate(images_thresh):
        fn, c_img, t_img = data['filename'], data['clahe_img'], data['thresh_img']
        print( 'Processing File: {}'.format(fn))

        plt.subplot(121)
        plt.imshow(c_img, cmap='gray')
        plt.title('Original')
        plt.subplot(122)
        plt.imshow(t_img, cmap='gray')
        plt.title('Binary Threshold')
        plt.show()
        plt.pause(0.1)

        top_regions = np.argpartition(data['count'], -2)[-2:]
        print(len(top_regions))
        top_counts = data['count'][top_regions]
        print ('Top region pixel counts: {}'.format(top_counts))
        my_mask = mask( t_img, data['labels'], region=top_regions[1])
        image = c_img * my_mask

        image = np.array(image, dtype = np.uint8)
```

```
                plt.imshow(image, cmap='gray')
                plt.title(fn)
                plt.show()
                plt.pause(0.1)

                input4 = input("Save post processed image (Y/N): ").lower()
                if input4 == 'y':
                    save(fn, image)

                clear_output()
            return revist
```

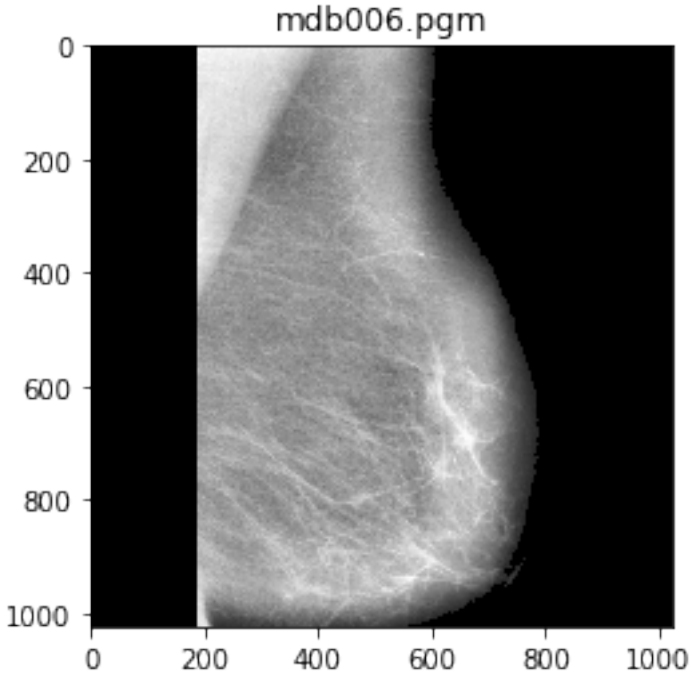
In [36]:
```
#save the pre-processed images

#add the pre-processed images directory
SAVE_DIR = 'C:/Users/victory/Desktop/MIAS/Pre-processed'

def save(fn, img, location=SAVE_DIR):
    print( 'Saving: {}'.format(location + fn))
    cv2.imwrite(location + fn, img)
    time.sleep(2)
```

In [ ]:
```
#applying all the previous methods on the dataset

filenames = [ filename for filename in os.listdir('C:/Users/victory/Desktop/MIAS')
              if filename.endswith('.pgm')] #raw images directory

clahe_images = []
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
os.chdir('C:/Users/victory/Desktop/MIAS/') #raw imges directory
for filename in filenames:
    img = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    clahe_images.append({'filename': filename, 'clahe_img': clahe.apply(img)})
os.chdir(os.getcwd()) #current directory

images_thresh = threshold(clahe_images)
print (len(images_thresh))

remaining = clean_art(images_thresh)
remaining_fn = [item['filename'] for item in remaining]
```

```
Processing File: mdb006.pgm
```



2
Top region pixel counts: [509627 526422]



May 8, 2019

In [1]:
```
import os, urllib
from urllib.request import urlretrieve
def download(url):
    filename = url.split("/")[-1]
    if not os.path.exists(filename):
        urlretrieve(url, filename)
def get_model(prefix, epoch):
    download(prefix+'-symbol.json')
    download(prefix+'-%04d.params' % (epoch,))

get_model('http://data.mxnet.io/models/imagenet/resnet/152-layers/resnet-152', 0)
```

In [2]:
```
import mxnet as mx
sym, arg_params, aux_params = mx.model.load_checkpoint('resnet-152', 0)
```

In [3]:
```
mod = mx.mod.Module(symbol=sym, context=mx.cpu())
```

In [4]:
```
mod.bind(for_training = False,
         data_shapes=[('data', (1,3,224,224))])
mod.set_params(arg_params, aux_params)
```

In [5]:
```
import numpy as np
import cv2

from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
from keras.applications.vgg16 import VGG16

def get_image(filename):
    img = cv2.imread(filename)  # read image in b,g,r order
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)   # change to r,g,b order
    img = cv2.resize(img, (224, 224))  # resize to 224*224 to fit model
    img = np.swapaxes(img, 0, 2)
    img = np.swapaxes(img, 1, 2)  # change to (channel, height, width)
    img = img[np.newaxis, :]  # extend to (example, channel, heigth, width)
    return img
```

In [6]:
```
from collections import namedtuple
Batch = namedtuple('Batch', ['data'])
```

In [7]:
```
all_layers = sym.get_internals()
all_layers.list_outputs()[-10:-1]
```

Out[7]:
```
['bn1_moving_var',
 'bn1_output',
 'relu1_output',
 'pool1_output',
 'flatten0_output',
 'fc1_weight',
 'fc1_bias',
 'fc1_output',
 'softmax_label']
```

In [8]:
```
all_layers = sym.get_internals()
sym3 = all_layers['flatten0_output']
mod3 = mx.mod.Module(symbol=sym3, context=mx.cpu())
mod3.bind(for_training=False, data_shapes=[('data', (1,3,224,224))]) #batch size = 1 aw
mod3.set_params(arg_params, aux_params)

#sym3.list_arguments()
```

In [9]:
```
#extraction of features of our classes
from PIL import Image
import glob

out=[]

data_pos_mal = np.array([get_image
                         ('C:/Users/victory/Desktop/MIAS/ClassMalignant/MalignantROI/'+i
                         for img in os.listdir('C:/Users/victory/Desktop/MIAS/ClassMalig
                         if img.endswith('.jpg')])
data_pos_ben = np.array([get_image
                         ('C:/Users/victory/Desktop/MIAS/ClassBenign/BenignROInew/'+img)
                         for img in os.listdir('C:/Users/victory/Desktop/MIAS/ClassBenig
                         if img.endswith('.jpg')])
data_neg = np.array([get_image
                     ('C:/Users/victory/Desktop/MIAS/ClassNormal/NormalROINew/'+img)
                     for img in os.listdir('C:/Users/victory/Desktop/MIAS/ClassNormal/No
                     if img.endswith('.jpg')])

data_pos= np.append(data_pos_mal,data_pos_ben,axis=0)
X = np.append(data_pos,data_neg,axis=0)

print (X.shape)
print (data_pos.shape)
print (data_neg.shape)
```

```
(504, 1, 3, 224, 224)
(91, 1, 3, 224, 224)
```

```
(413, 1, 3, 224, 224)
```

```
In [10]: for img in X:
             mod3.forward(Batch([mx.nd.array(img)]))
             out.append(mod3.get_outputs()[0].asnumpy())
         out = np.array(out)

         print (out.shape)
         X = out
         X = X.reshape((X.shape[0], -1), order='F')
         print (X.shape)
```

```
(504, 1, 2048)
(504, 2048)
```

```
In [13]: Y = np.ones(504) #Alllabels
         for i in range(len(Y)):
             if i >= 91: #positive size
                 Y[i]= 0
```

```
In [14]: from sklearn.feature_selection import SelectKBest
         from sklearn.feature_selection import chi2

         # feature extraction
         test = SelectKBest(score_func=chi2, k=10)
         fit = test.fit(X, Y)
```

```
In [15]: # summarize scores
         np.set_printoptions(precision=3)
         print(fit.scores_)
         features = fit.transform(X)
         # summarize selected features
         print(features[0:5,:])

         print(features.shape)
```

```
[18.17  19.053  5.989 ... 21.732  1.312  2.641]
[[7.203 1.986 3.525 9.792 2.749 4.66  3.674 2.454 2.287 2.913]
 [1.558 0.356 2.267 2.361 1.933 0.556 1.31  1.058 4.486 2.762]
 [6.194 1.458 3.683 8.012 3.1   3.684 2.807 2.123 4.    3.927]
 [6.771 1.508 3.618 9.606 3.006 4.487 3.428 2.657 3.028 3.145]
 [2.766 0.783 2.619 4.029 2.067 1.578 1.349 1.703 3.869 2.925]]
(504, 10)
```

```
In [16]: from sklearn.decomposition import PCA
```

```
        # feature extraction
        pca = PCA(n_components=10)
        fit = pca.fit(X)
        features = fit.transform(X)

        # summarize components
        print("Explained Variance: ", fit.explained_variance_ratio_)
        print(fit.components_)
```

```
Explained Variance:  [0.247 0.131 0.087 0.071 0.045 0.038 0.035 0.028 0.024 0.02 ]
[[-0.012 -0.021 -0.004 ...  0.055 -0.006  0.018]
 [ 0.021 -0.002  0.029 ...  0.036 -0.001  0.021]
 [-0.003 -0.005 -0.027 ...  0.045 -0.011  0.03 ]
 ...
 [-0.    -0.019 -0.017 ...  0.015 -0.    -0.01 ]
 [-0.002 -0.022 -0.037 ... -0.039  0.002 -0.001]
 [-0.007 -0.018  0.007 ...  0.058 -0.01  -0.041]]
```

```
In [21]: from sklearn.model_selection import train_test_split

        imgs_train, imgs_test, input_gts_train, input_gts_test, =
            train_test_split(features , Y, test_size=0.3, random_state=42)
```

```
In [22]: from sklearn.ensemble import RandomForestClassifier
        split =3
        dep = 140

        Mrfc = RandomForestClassifier(n_estimators = 1000,
                                      bootstrap = True,
                                      oob_score = True,
                                      criterion = 'gini',
                                      max_features = 'auto',
                                      class_weight="balanced" ,
                                      max_depth = dep,
                                      min_samples_split = int(10000 / split),
                                      min_samples_leaf = int(3000 / split),
                                      max_leaf_nodes = None,
                                      n_jobs=-1
                                     )

        Mrfc.fit(imgs_train,input_gts_train)
        input_gts_train0 = Mrfc.predict(imgs_train)
        accuray_train = np.mean((input_gts_train0 - input_gts_train) ** 2)
        print ('split is ' + str(split) + ' depth is ' + str(dep))
        print ('train')
        print(accuray_train, Mrfc.score(imgs_train,input_gts_train))
        input_gts_test0 = Mrfc.predict(imgs_test)
```

```
        accuray_test = np.mean((input_gts_test0 - input_gts_test) ** 2)
        print ('test')
        print(accuray_test, Mrfc.score(imgs_test,input_gts_test))
```

```
split is 3 depth is 140
train
0.1534090909090909 0.8465909090909091
test
0.24342105263157895 0.756578947368421
```

In [23]:
```
        from sklearn import tree
        from sklearn.model_selection import cross_val_score
        #classifier
        clf = tree.DecisionTreeClassifier()

        clf = clf.fit(X, Y)

        cv_scores = cross_val_score(clf, X, Y, cv=10)

        print(cv_scores)
        cv_scores.mean()
```

```
[0.962 0.98  0.961 0.9   1.    1.    0.98  1.    0.98  0.94 ]
```

Out[23]:
```
0.9702714932126696
```

In [24]:
```
        print(Mrfc.feature_importances_) #.mean()
        feature_importance_sum = np.sum(clf.feature_importances_)
        print(feature_importance_sum)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
1.0
```

In [25]:
```
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.preprocessing import StandardScaler
        from sklearn import model_selection

        scaler = StandardScaler()
        scaler = scaler.fit(imgs_train)
        x_train = scaler.transform(imgs_train)
        x_test = scaler.transform(imgs_test)
        clf = KNeighborsClassifier(n_neighbors=5)
        clf = clf.fit(imgs_train, input_gts_train)

        score = clf.score(imgs_test, input_gts_test)
        scores = model_selection.cross_val_score(clf, X, Y, cv=10)
        print('K-Nearest Neighbor accuracy: ' + str(round(scores.mean()*100, 3)) + '%')
```

```
K-Nearest Neighbor accuracy: 96.219%
```

```
In [26]: from sklearn.naive_bayes import GaussianNB


         clf = GaussianNB()
         clf = clf.fit(imgs_train, input_gts_train)
         scores = model_selection.cross_val_score(clf, X, Y, cv=10)
         print('Naive Bayes accuracy: ' + str(round(scores.mean()*100, 3)) + '%')
```

```
Naive Bayes accuracy: 94.65%
```

```
In [27]: from sklearn import svm

         C = 1.0
         svc = svm.SVC(kernel='linear', C=C).fit(imgs_train, input_gts_train)
         scores = model_selection.cross_val_score(svc, X, Y, cv=10)
         print('Support Vector Machine accuracy: ' + str(round(scores.mean()*100, 3)) + '%')
```

```
Support Vector Machine accuracy: 98.412%
```

```
In [28]: from sklearn.linear_model import LogisticRegression

         lgr = LogisticRegression()
         lgr = lgr.fit(imgs_train, input_gts_train)
         scores = model_selection.cross_val_score(lgr, X, Y, cv=10)
         print('Logistic Regression accuracy: ' + str(round(scores.mean()*100, 3)) + '%')
```

```
Logistic Regression accuracy: 98.612%
```

```
In [29]: from sklearn.model_selection import RandomizedSearchCV
         # number of trees in random forest
         n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
         # number of features at every split
         max_features = ['auto', 'sqrt']

         # max depth
         max_depth = [int(x) for x in np.linspace(100, 500, num = 11)]
         max_depth.append(None)
         # create random grid
         random_grid = {
          'n_estimators': n_estimators,
          'max_features': max_features,
          'max_depth': max_depth
          }
```

```
        # Random search of parameters
        rfc_random = RandomizedSearchCV(estimator = Mrfc, param_distributions =
            random_grid, n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -1)
        # Fit the model
        rfc_random.fit(imgs_train, input_gts_train)
        # print results
        print(rfc_random.best_params_)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits


[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:   54.7s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:  3.4min
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:  6.3min finished


{'n_estimators': 1000, 'max_features': 'auto', 'max_depth': 140}
```

```
In [30]: from sklearn.model_selection import cross_val_score
         from sklearn.metrics import classification_report, confusion_matrix


         rfc_cv_score = cross_val_score(Mrfc, X, Y, cv=10, scoring='roc_auc') #cv=10
```

```
In [31]: print("=== Confusion Matrix ===")
         print(confusion_matrix(input_gts_test, input_gts_test0))
         print('\n')
         print("=== Classification Report ===")
         print(classification_report(input_gts_test, input_gts_test0))
         print('\n')
         print("=== All AUC Scores ===")
         print(rfc_cv_score)
         print('\n')
         print("=== Mean AUC Score ===")
         print("Mean AUC Score - Random Forest: ", rfc_cv_score.mean())
```

```
=== Confusion Matrix ===
[[115   0]
 [ 37   0]]


=== Classification Report ===
             precision    recall  f1-score   support

        0.0       0.76      1.00      0.86       115
        1.0       0.00      0.00      0.00        37

avg / total       0.57      0.76      0.65       152
```

```
=== All AUC Scores ===
[0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5]


=== Mean AUC Score ===
Mean AUC Score - Random Forest:  0.5


In [33]: from sklearn import metrics
         (precision, recall, fscore, support) =
         metrics.precision_recall_fscore_support(input_gts_test,
                                                 input_gts_test0,
                                                 average='weighted',
                                                 warn_for=tuple())
         print(precision, recall, fscore, support)

0.5724117036011079 0.756578947368421 0.6517346737630593 None
```

Bibliography

[1] A. Likas A. Papadopoulos, D.I. Fotiadisb. Characterization of clustered microcalcifications in digitized mammograms using neural networks and support vector machines. *Artificial Intelligence in Medicine*, 2004.

[2] Said Jai-Andaloussi Abderrahim Sekkaki Abdelali Elmoufidi, Khalid El Fahssi. Detection of regions of interest in mammograms by using local binary pattern and dynamic k-means algorithm. *International Journal of Image and Video Processing: Theory and Application*, pages 11–18, 2014.

[3] Nystrm L Njor S Jonsson H Paap E Massat N Duffy S Lynge E Paci E Broeders M, Moss S. The impact of mammographic screening on breast cancer mortality in europe: a review of observational studies. *SAGE Journal*, 19:14–25, 2012.

[4] KangTsung Chang. Computation for bilinear interpolation. *Introduction to Geographic Information Systems*, 5th ed, 2009.

[5] Alan Williams Katy Wolstencroft Chris Taylor Chris Rose, Daniele Turi. Web services for the ddsm and digital mammography research. *IWDM*, 2006.

[6] Arzav Jain Daniel Lvy. Breast mass classification from mammograms using deep convolutional neural networks. In *arXiv:1612.00542v1*. Stanford University, 2016.

[7] Xiuzhen Cheng Dechang Chen. A simple implementation of the stochastic discrimination for pattern recognition. *F.J. Ferri et al. (Eds.)*, 1:882887, 2000.

[8] Zsuzsa Unger Peter Pollner Istvan Csabai Dezso Ribli, Anna Horvath. Detecting and classifying lesions in mammograms with deep learning. 2012.

[9] Mohamed Fayez Enas Mostafa. Double blind reading. PaxeraMed, 2018.

[10] J Suckling et al. The mammographic image analysis society digital mammogram database exerpta medica. *International Congress Series*, 1069:375–378, 1994.

[11] Bradley M. Hemminger Marla Deluca R. Eugene Johnston Keith Muller M. Patricia Braeuning Etta D. Pisano, Shuquan Zong and Stephen M. Pizer. Contrast limited adaptive histogram equalization image processing to improve the detection of simulated spiculations in dense mammograms. *Journal of Digital Imaging*, 11(4):193–200, 1998.

[12] Shaode Yu Yaoqin Xie Fan Jiang, Hui Liu. Breast mass lesion classification in mammograms by transfer learning. 2017.

[13] Rohith Gandhi. Support vector machine - introduction to machine learning algorithms. *Towards Data Science*, 2018.

[14] Emily F. Conant Susan P. Weinstein Harold L. Kundel, Calvin F. Nodine. Holistic component of image perception in mammogram interpretation. *Radiology: Volume 242: Number 2*, 2007.

[15] Tin Kam Ho. Random decision forests. *ATT Bell Laboratories*, 1995.

[16] Aaron Courville Ian Goodfellow, Yoshua Bengio. Deep learning. In *Deep Learning*. MIT, 2016.

[17] I.T. Jolliffe. Principal component analysis, second edition. 2002.

[18] Xiaoming Liu Jun Liu, Jianxun Chen and J Tang. An investigate of mass diagnosis in mammogram with random forest. *Fourth International Workshop on Advanced Computational Intelligence*, 2011.

[19] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. *Microsoft Research*, 2015.

[20] Andrew Zisserman Karen Simonyan. Very deep convolutional networks for large-scale image recognition. *University of Oxford*, 2015.

[21] E. M. Kleinberg. Stochastic discrimination. *Annals of Mathematics and Artificial Intelligence*, 1:207239, 1990.

[22] Ender Konukoglu Lukas Jendele, Anton S. Becker. Adversarial augmentation for enhancing classification of mammography images. 2019.

[23] Gromet M. Comparison of computer-aided detection to double reading of screening mammograms. *AJR Am J Roentgenol*, 190(4):854– 863, 2008.

[24] Sue Astley Chris Taylor Michael Berks, Zezhi Chen. Detecting and classifying linear structures in mammograms using random forests. 2011.

[25] Daniel C Moura Miguel Lopez, Naimy Gonzlez Posada. Bcdr: A breast cancer digital repository. *15th International Conference on Experimental Mechanics*, 2012.

[26] Yide Ma Yanan Guo Yurun Ma Keju Wang Min Dong, Xiangyu Lu. An efficient approach for automated mass segmentation and classification in mammograms. In *J Digit Imaging*, pages 613–625. Society for Imaging Informatics in Medicine, 2015.

[27] Al Hussein Ahmed Mohammed A.-M. Salem. Mammogram-based cancer detection using deep convolutional neural networks. 2018.

[28] Domingues I Cardoso A Cardoso MJ Cardoso JS Moreira IC, Amaral I. Inbreast: toward a full-field digital mammographic database. *Acad Radiol*, 19(2), 2012.

[29] Yang Dan Natalia Caporale. Spike timingdependent plasticity: A hebbian learning rule. *Annual Review of Neuroscience*, 31, 2008.

[30] Andrew P. Bradley Neeraj Dhungel, Gustavo Carneiro. Automated mass detection from mammograms using deep learning and random forest. 2015.

[31] Jakob Nielsen. Powers of 10: Time scales in user experience. 2009.

[32] R. Woods R. Gonzalez. Digital image processing. volume 2. Prentice Hall, 2002.

[33] Danilo Pereira Rodrigo Ramos, Marcelo Nascimento. Texture extraction: An evaluation of ridgelet, wavelet and co-occurrence based methods applied to mammograms. *Expert Systems with Applications*, 39:11036–11047, 2012.

[34] S. Arumugam S. Usha. Calcification classification in mammogram using decision trees. *International Journal of Computer and Information Engineering*, 9(9), 2015.

[35] Rong Ge Tengyu Ma Sanjeev Arora, Aditya Bhaskara. Provable bounds for learning some deep representations. *abs/1310.6343*, 2013.

[36] saranya Mandava. Cross validation and hyperparameter tuning in python. 2018.

[37] Srinivasan Emperumal Saraswathi Duraisamy. Computer-aided mammogram diagnosis system using deep learning convolutional fully complex-valued relaxation neural network classifier. *IET Journals*, 11:656–662, 2017.

[38] Christian Szegedy Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

[39] Ling Guan Songyang Yu. A cad system for the automatic detection of clustered microcalcifications in digitized mammogram films. *Transactions On Medical Imaging*, 19(2), 2000.

[40] Arnau Oliver Robert Mart Xavier L. Jordi Freixenet Joan C. Vilanova Desire Sidibe Fabrice Meriaudeau Soumya Ghose1, Jhimli Mitra. A random forest based classification approach to prostate segmentation in mri. 2011.

[41] Caglar Subasi. Logistic regression classifier. *Towards Data Science*, 2019.

[42] Massimiliano Pontil Theodoros Evgeniou. Support vector machines: Theory and applications. *Machine Learning and Its Applications, Advanced Lectures*, 2001.

[43] Pranaw K P Deepa Shenoy Venugopal K R L M Patnaik Vibha L, Harshavardhan G M. Classification of mammograms using decision trees. *International Database Engineering and Applications Symposium*, 10, 2006.

[44] Kaiming He Xiangyu Zhang, Jianhua Zou and Jian Sun. Accelerating very deep convolutional networks for classification and detection. 2015.