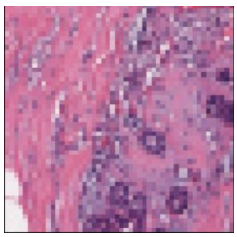
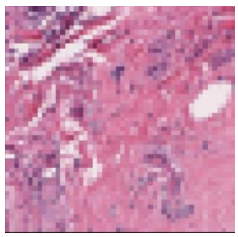
Machine Learning Engineer Nanodegree

Rapid Diagnosis of IDC in Mammography Scans using Deep Learning



Benign
(IDC Negative)

Capstone Project
Liad Levi-Raz
Sep, 2018



Malignant
(IDC Positive)

Contents

I. Definition

Breast Cancer

"Breast cancer is the most common cancer in women and it is the main cause of death from cancer among women in the world." (1)

According to breastcancer.org (2) :

"About **1 in 8** U.S. women (about 12.4%) will develop invasive breast cancer over the course of her lifetime."

"In 2018, an estimated **266,120 new cases** of invasive breast cancer are expected to be diagnosed in women in the U.S., along with **63,960** new cases of non-invasive (in situ) breast cancer."

About **40,920 women** in the U.S. are expected **to die** in 2018 from breast cancer, though death rates have been decreasing since 1989... These decreases are thought to be the result of treatment advances, **earlier detection** through screening, and increased awareness."

"Besides skin cancer, breast cancer is the **most commonly diagnosed** cancer among American women. In 2017, it's estimated that about 30% of newly diagnosed cancers in women will be breast cancers." (2)

Early Detection

An **early detection** and treatment of breast cancer at an early stage can save lives. Cancer that's diagnosed at an early stage, before it's had the chance to get too big or spread is more likely to be treated successfully. If the cancer has spread, treatment becomes more difficult, and generally a person's chances of surviving are much lower.

For example: according to Cancer Research UK (3), more than 90% of women diagnosed with breast cancer at the earliest stage, survive their disease for at least 5 years, compared to around 15% survival rate, for women diagnosed with the most advanced stage of the disease.

Invasive Ductal Carcinoma (IDC)

An **Invasive Ductal Carcinoma** is the most common form of breast cancer, according to John Hopkins Medicine (4) 80% of all breast cancer diagnoses are IDC. This cancer subtype is a cancer that began growing in a milk duct and has invaded other breast tissues outside of the duct.

Diagnosis

There are a few methods to diagnose IDC breast cancer:

- Screening (Digital) mammography, Ultrasound, MRI, Biopsy, Pathology

Screening mammography is usually the first diagnosis step (except for self-checking), it is a process of examining a human breast using low-energy X-rays. X-ray images are captured from 2 angles of each breast. It has been shown to reduce breast cancer mortality by 38-48% among patients.

These images are inspected for malignant lesions by **one or two experienced radiologists**. Suspicious cases are called back for further diagnostic evaluation… the manual diagnosis of Mammography has a relatively high **false-negative** rate (**missed cancer**) of at least 10% (1)

The diagnosis process is described by radiologists as "monotonous, tiring, lengthy, costly and most importantly, prone to errors" (5)

In most cases where the patient is not in a risk group the results, the results will be sent in a letter within **30 days** saying the results were normal, if there are suspicious findings – it may still take **5 days** to get a result (and be called back for additional tests) (6)

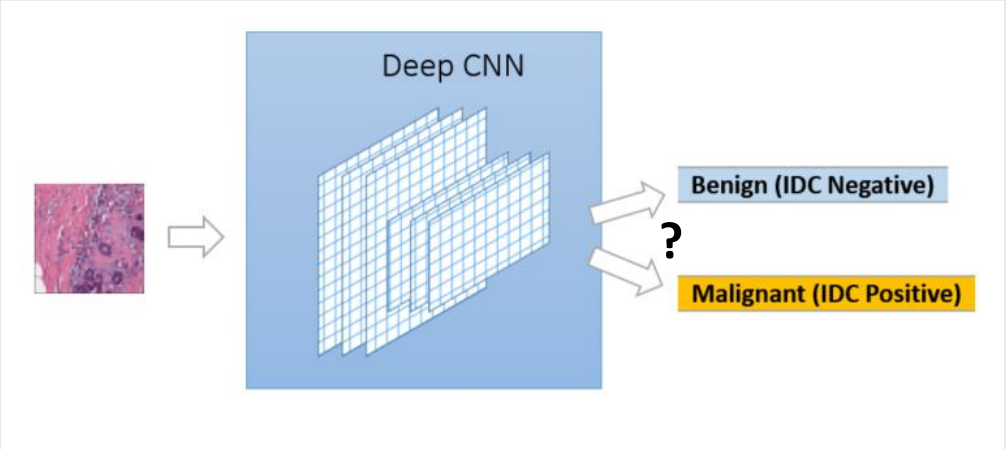
I am personally familiar with a few cases of breast cancer in women – unfortunately, most of them eventually did not end well.

Therefore, I am motivated to practice and hopefully improve the early detection of cancer and the accuracy of the diagnosis. Getting more accurate results quicker, may reduce the stress and anxiety for women that going through the screening. This is especially stressful for women that need to live with the fear that breast cancer may reoccur.

I was inspired (and impressed…) by the following medical publication: "Detecting and classifying lesions in mammograms with Deep Learning" - published on: 15 March 2018(5)

Problem Statement

The challenge is to train a CNN that will receive the labeled digital Mammography images as input (benign or malignant) and will learn to predict whether a given image is benign or malignant.



The problem I want to concentrate on, is automating the manual Mammography scans diagnosis process, and I hope to get 'extra benefits' by doing so:

1. Provide immediate diagnosis results (the vision is to make it as simple as using Dr. McCoy's scanner in Star Trek)
2. Matching or improving the quality of the diagnosis – reduce the **false-negative rate** (~beat 85% F1 score or reduce false-negative rate from 10%) – more details on the F1 metric and others in the following 'Evaluation Metrics' section

As mentioned in the previous section, the diagnosis process is a manual process that sometimes take **days**, because of the high load on expert radiologists that analyze the results, and being a manual process which involves scanning digital images – it is also **error prone** (results accuracy is considered to be between ~83% to 87% depending on age, history, and other parameters ref: 7, 8, 9)

Using Deep Learning to predict IDC can provide immediate and sometimes more accurate results.

Evaluation Metrics

Evaluating the performance of the unbalanced dataset (71.6% non-IDC and 28.4% IDC), will required using the F1 score (10) which is based on the precision and recall, this is in order to avoid the accuracy paradox (20)

In every epoch the F1 score is calculated for every class based on the precision and recall.

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted **positive** observations. The question that this metric answer is: of all patients labeled as IDC(+) or IDC(-), how many are actually IDC(+) or IDC(-)?

Precision = TP / TP + FP

Recall (**Sensitivity**) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class. The question recall answers is: Of all the patients that actually IDC(+) or IDC(-), how many did we label as IDC(+) or IDC(-)

Recall = TP/TP+FN

"**F1 score** - F1 Score is the **weighted average** of **Precision** and **Recall**. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall."(11)

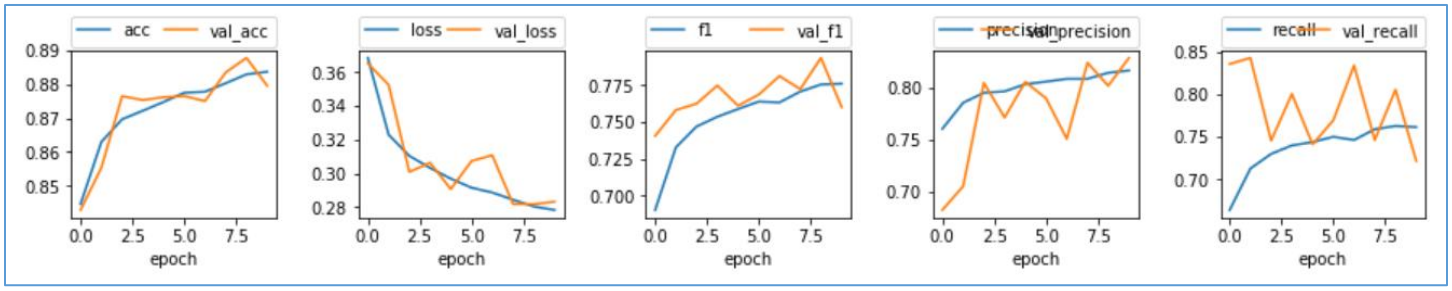
F1 Score = 2 * (Recall Precision) / (Recall + Precision)

(Note: KFold is used as a cross valuation method)

For the **Full CNN** at the end of the **training**, the following information is presented:

Training

1. Training Learning curves:
    - Per fold -three graphs displaying F1, Recall, Precision, Accuracy and Loss curves
    - Per fold summary for the same metrics
2. Expected results:
    - The loss and val_loss are both decreasing
    - The f1 and val_f1 are both increasing



Evaluation

3. At the end of the **model evaluation** the following information is presented:
    - Summary of F1, Recall, Precision, Accuracy and Loss scores
    - Classification Report across predicted classes
    - Confusion Matrix visualization
4. Expected results:
    - A good F1 score (> 85%)
    - Classification Report: A good balanced F1 score on the
    - Confusion matrix: High scores on the Top Left and Bottom Right cells (darker colors)

For the **benchmark** the expected results are different – I expect to see that the model performed badly, predicting a single class only, for example: all images are IDC(-) , this will be shown in the f1 score, classification report and confusion matrix – see 'Benchmark' section.

II. Analysis

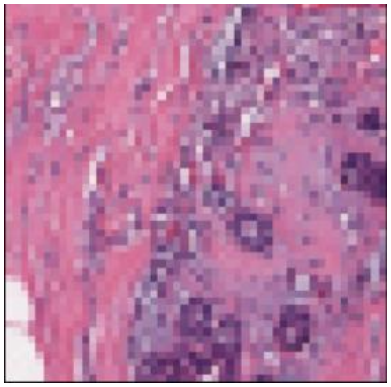
Data Exploration

The selected dataset is the "Breast Histopathology Images" found on Kaggle:

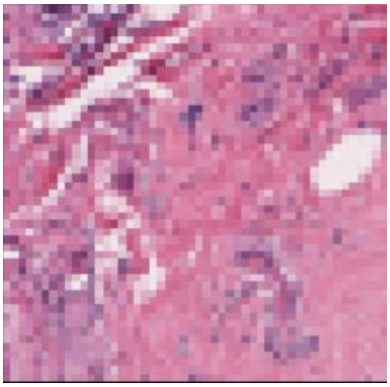
https://www.kaggle.com/paultimothymooney/breast-histopathology-images (12)

The dataset contains folders with the digital mammography images, **198,738 IDC negative** images (71.6%) and **78,786 IDC positive** images (28.4%), which are going to be used to train (+ validate) and test the CNN.

Sample images:

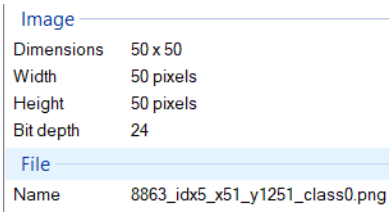


Benign (IDC Negative)



Malignant (IDC Positive)

The images sizes are: 50 X 50 pixels and 24 color depth



“Pathologists typically focus on the regions which contain the IDC. As a result, one of the common pre-processing steps for automatic aggressiveness grading is to delineate the exact regions of IDC inside of a whole mount slide. The original dataset consisted of 162 whole mount slide images of Breast Cancer (BCa) specimens scanned at 40x. From that, 277,524 patches of **size 50 x 50** were extracted (198,738 IDC negative and 78,786 IDC positive).” - Reference from Kaggle (12).

The images file name format has the following name pattern, (an underscore character ”_” is used as a separator):

File name	Patient ID	Original X-coordinate where the patch was cropped from	Original Y-coordinate where the patch was cropped from	The labeled IDC class 0 - non IDC 1 - IDC
10253_idx5_x1351_y1101_class0.png	10253_idx5	1351	1101	0

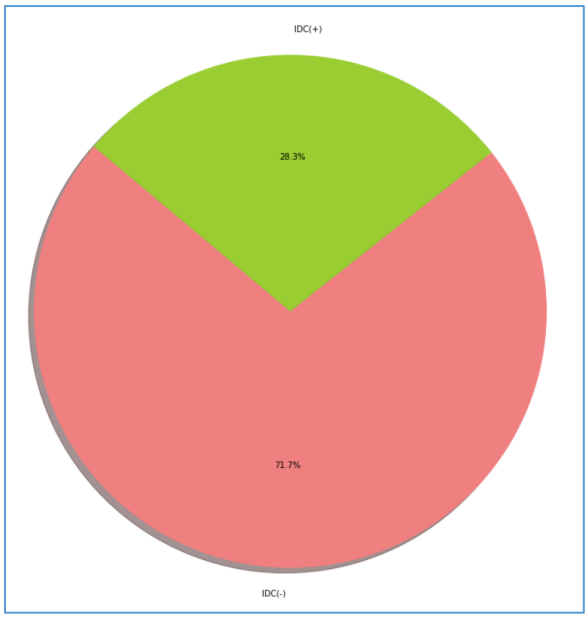
The dataset was divided into two subsets:

3. Training - 256396 images (the training dataset will be further divided using KFold cross validation)
4. Testing - 21128 image – kept aside for final model validation

Data Imbalance

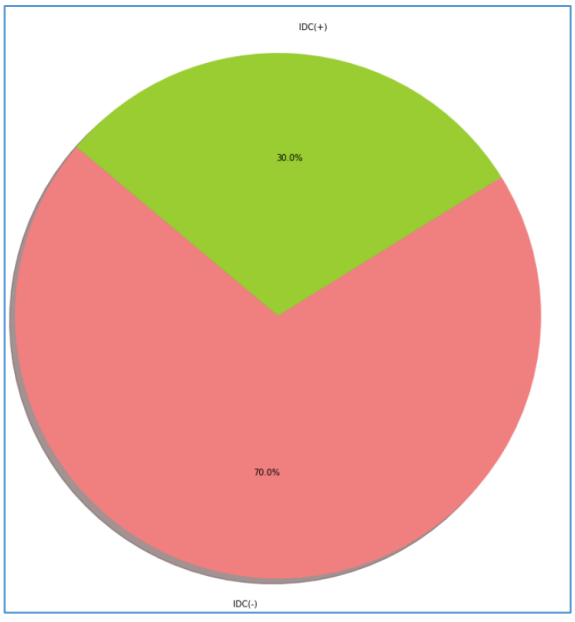
The dataset description shows that dataset classes are unbalanced; around 70% IDC Negative, and around 30% are IDS Positive. This is also confirmed by the EDA as shown in the following pie charts for Training and Testing datasets:

Training dataset class balance



71.7% IDC(-) / 28.3% IDC(+)

Testing dataset



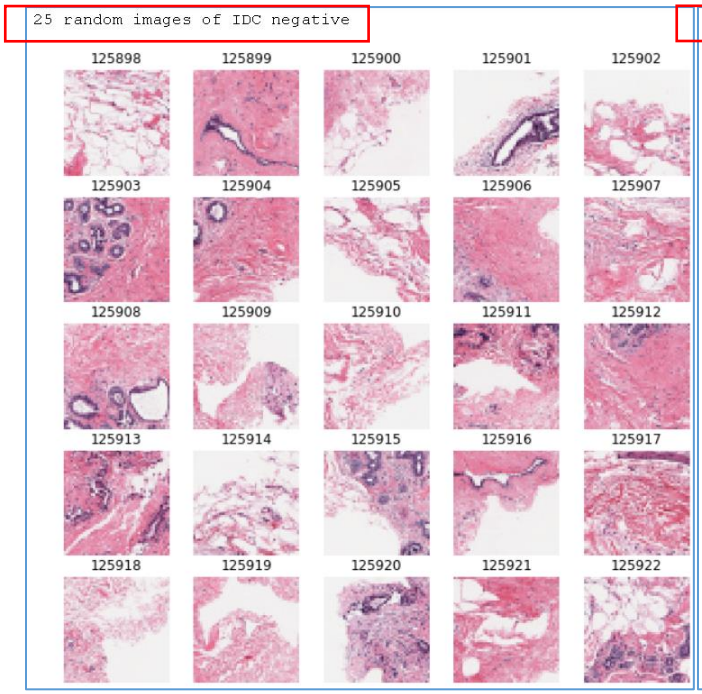
70.0% IDC(-) / 30.0% IDC(+)

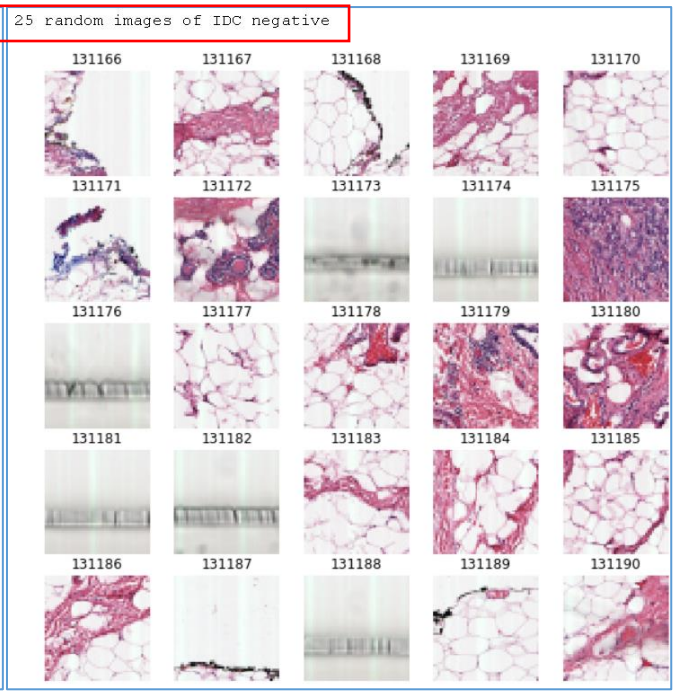
I am going to relate to this fact during training, when fitting the data, because data imbalance is known to affect the performance of ML models.

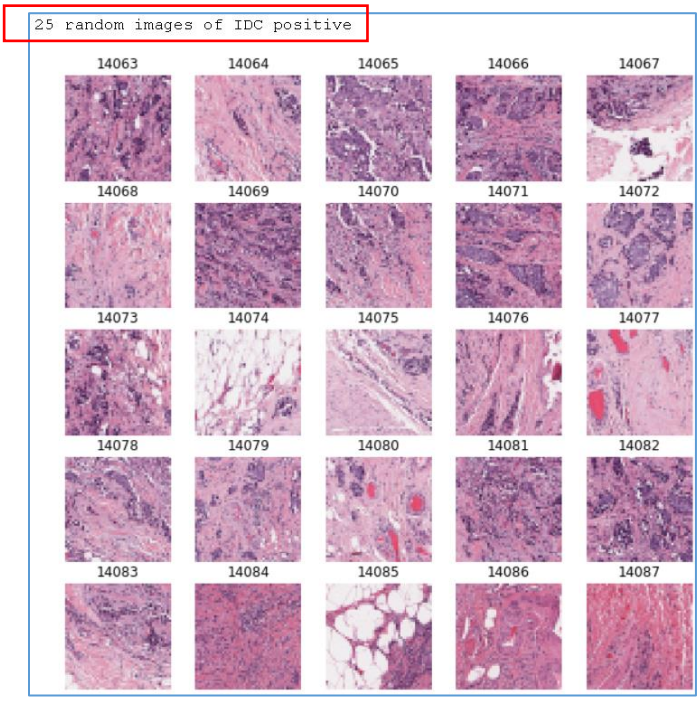
Exploratory Visualization

Here are three plots, of 25 images each, selected randomly:

The first and second groups of 25 images are labelled as IDC Negative (Benign) and the third group of 25 images are labelled as IDC Positive (Malignant).







While examining the input images, I could not identify any **shape** or **color** pattern that can clearly classify images as IDC(-) or IDC(+), the 'marble-like' shapes are quite similar to the untrained eye.

However, a few outliers were identified (second group grayscale images), they seem to be invalid scans.Here is an example of an outlier, it seems to be an invalid scan :

I didn't find outliers while reviewing the **IDC (+)** images (maybe the dataset was properly cleaned of such). Anyway, these outliers should only add some 'noise' to the training data, I don't expect they will damage the CNN prediction quality.

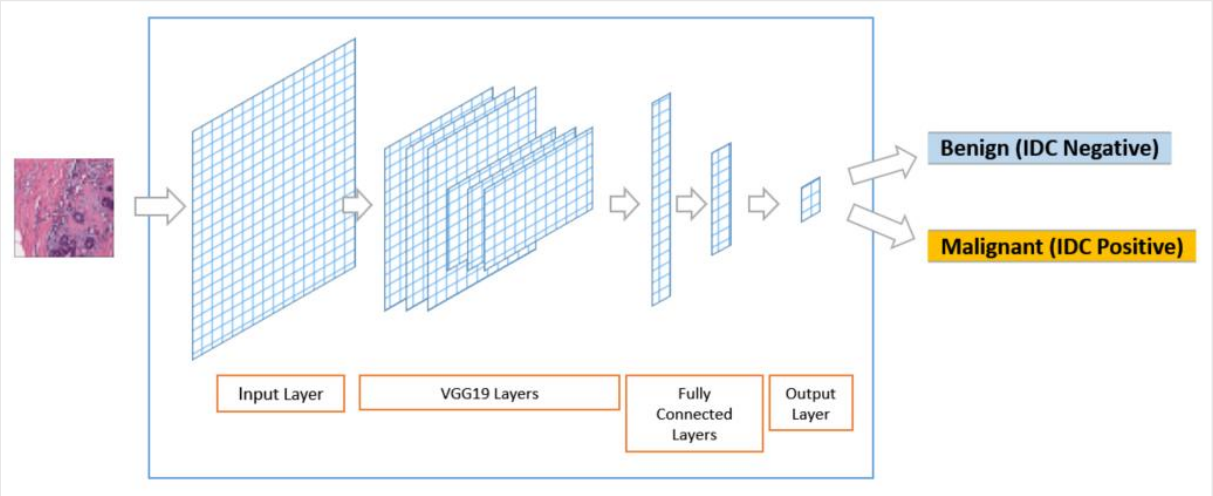
Algorithms and Techniques

Based on the information I gathered about the domain and the problem, and based on my personal knowledge, I believe that the approach of using Deep Learning and Convolutional Neural Network(CNN), will prove itself worthy.

"In the last two decades, Computer Aided Detection (CAD) systems were developed to help radiologists analyze screening mammograms, however benefits of current CAD technologies appear to be contradictory, therefore they should be improved to be ultimately considered useful**. Since 2012, deep convolutional neural networks (CNN) have been a tremendous success in image recognition, reaching human performance.** These methods have greatly surpassed the traditional approaches, which are similar to currently used CAD solutions. Deep CNN-s have the potential to revolutionize medical image analysis." ("Detecting and classifying lesions in mammograms with Deep Learning" - published on: 15 March 2018) (5)

Therefor I designed and trained a CNN for solving the problem using Keras (13).

Training a CNN from scratch might have been useful, but it would have taken many resources and a long time to train and tune the CNN, so my approach is based on a Transfer Learning (14).



Transfer Learning

Having a relatively large dataset (~250K) I chose to use **Transfer Learning** from a VGG19 CNN pretrained on 'imagenet' (available as a Keras application) (15).

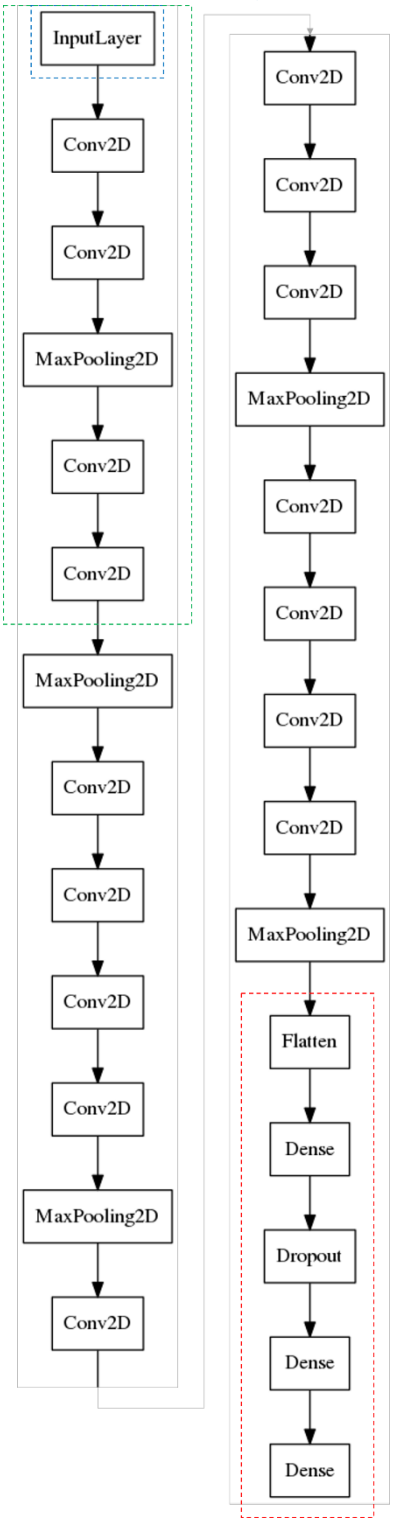
While training the CNN I also tried Transfer Learning from other pretrained CNN architectures such as VGG16, MobileNet and NASNet – as they were the ones accepting the dataset images in their original size (I didn't want to rescale the images and risk corrupting the data)

Looking at benchmarks that were performed on the Keras applications it seems that VGG19 is just right for the task, it has 26 original layers (not too many…) and performed quite well; 91% in the Top-5 accuracy.

The plan was to load the pretrained VGG19 CNN without its 'top-layer' which was trained to classify the 'imagenet' data, instead, to add my own top layer, with 2 Dense layers that will learn to classify IDC(-) and IDC(+).

The new CNN architecture is displayed in the following diagram:

CNN Architecture Diagram



There are total of 27 layers (the original VGG19 has 28 layers)

The input layer, marked by blue dashed frame, accepts images with an input shape of 50 X 50 X 3

The next 21 layers is the original set of VGG19 Convolutional layers with 'relu' activation, and Max pooling.

The original 'top' of 6 layers was removed, and instead 5 fully connected layers was added to identify the specific IDC characteristics (marked by a red dashed frame):

2 fully connected Dense layers in size 128 + Dropout (0.4) to reduce overfitting (BatchNormalizarion and L2 Regularization were tried, but then removed, as I saw no significant effect on the final score)

The output layer is a fully connected dense layer with a Softmax activation that outputs 2 classes

The green dashed frame indicates the first 6 layers that were frozen during training, this is done to reuse the weights that were computed on 'imagenet', and also speeds up the training.

Cross validation using KFold

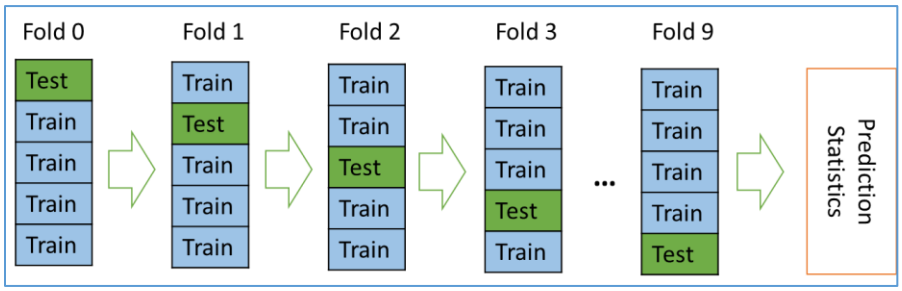
The purpose of using a cross validation technique is to evaluate how well the model generalizes during training.

Instead of splitting the training dataset once to 'train' and 'validation' subsets, I used Kfold in Keras (16), in which the training dataset is split into K-Folds (10 was selected after a few trials).

On every fold the training dataset is split into 10 ('K') subsets, 9 are used for training and 1 for validation.

This way the entire dataset is used for both training and validation, reducing the risk that some data will remain unseen during training, and also the risk of overfitting – it is a well-known technique for validating how well the model generalizes. at the end of all the folds training there is a prediction summary that takes an average of the scores achieved on every fold.

Here is an illustration of the KFold technique:



Benchmark

The purpose of the benchmark model is to show that the problem can actually be solved using machine learning and to provide a baseline score.

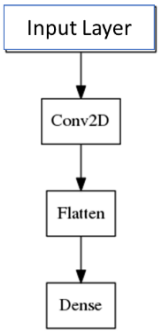
The benchmark model will run on the exact same dataset that will be used for training and will be measured using the same metrics (f1 score in this case), so the benchmark results are comparable to the 'final optimized' CNN model's results.

Since the dataset is unbalanced, I could use a naïve model based on the dataset distribution, saying that 71.6% non-IDC and 28.4% IDC – so for every new unseen image the model will predict that there is a 28.4% chance it will be IDC positive and 71.6% it will have non-IDC.

However, I chose to compare and present the results of a simple CNN as a benchmark model, which will be trained, validated and evaluated using the exact same dataset, and tuning.

Trying to train a very simple CNN to classify images is going to provide poor classification results, but any result that will either classify IDC(-) or IDC(+) images will do in this case, as it will show that the classification is possible.

The benchmark network architecture:



A very simple CNN with 1 Convolutional 2D layer (the Input) and 1 output Dense layer.

The Input layer is a Conv2D layer, with activation of 'relu', that accepts the images in shape 50X50X3

The output Dense layer is of size 2 (2 classes) with activation 'softmax' that matches a multiclass scenario, it outputs the probability of the each class of the two classes: IDC(-) and IDC(+)

III. Methodology

Data Preprocessing

The images in the input dataset are in a 'png' format, each in size 50 x 50 (see the Data Exploration section).

The dataset is imbalanced approx. 70% of the images are classified as IDC(-) and 30% as IDC(+).

As I want to build a multiclass classification model in Keras, I want to make sure all my images sizes and formats are aligned, the images are loaded as Numpy arrays, and the labels are One-Hot-Encoded.

In addition, I want to use image augmentation and shuffle to reduce the risk of overfitting to the training data.

The following table described the sequence of preprocessing steps performed on the input data to prepare it for the CNN training (and evaluation):

Preprocessing Step	Input	Output	Code reference / Function
1. *Load image file paths*	a directory path	a list with all the dataset images paths (*.png files)	*def loaded_training_images_paths(…)*
2. Load and resize images (align sizes)	a list of image paths	returns two Numpy arrays X - a matrix with the images data (2D) y - with the labels (1D)	*def images_to_nparray(…)* For every image path: -Read the image with OpenCV (17) (returns an RGB image array) -Resize the image to 50X50 (to align all images in case they are not 50X50)
3. On Hot Encoding labels – as I want to perform a multiclass classification (2 classes)	y	Y_hot – 2 arrays of hot encoded labels (2 classes)	*def run_training_evaluation(…)* … keras.utils.to_categorical()
4. Image Augmentation with Keras ImageDataGenerator (18)	X, y_hot		*def get_image_generators(…)*

Outliers: as explained in the Data Exploration section, some corrupted images were found on the training data – I chose not to remove them from the dataset as they were classified only in as IDC(-) - and there were only a few of them – I left them to play to role of some minor 'noise' and I don't expect them to affect the performance of the network.

Implementation

Working environment

Software	Version	
Python	3.6.5	
Jupyter Notebook Server	5.5.0	
GPU Server	Nvidia DGX01 with 8 Tesla V100 GPUs (19)	
Keras	2.2.2	
TensorFlow (tf-nightly-gpu)	1.11.0-dev20180808	
opencv-python	3.3.0.9	

Here is how I approached the problem:

1. Obtained the dataset from Kaggle
2. Manually split the dataset directory of 277524 images to:
   a. 256396 as a **training dataset** - approx. 70% - 30% class balance
   b. 21128 as a **test dataset** (unseen data kept aside) – approx. 70% - 30% class balance
3. Exploratory Data Analysis (EDA)including:
   a. Counting the total images, count images per class, calculate class weights (balance)
   b. Print samples of the dataset images
   c. Try to find outliers in the data just by looking at it
4. Metrics - based in the EDA findings:
   a. Selected F1 as an evaluation metric which is more suitable for unbalanced datasets
   b. Prepared helper visualization methods for visualizing the training and evaluation scores
5. Preprocessing including:
   a. Load the images and into Numpy arrays and resize them to 50px. X 50px. X 3 (RGB)
   b. Perform One Hot Encoding on the labels in to transform the 1D array to 2 arrays of labels per class – which is required for multi classification.
   c. Use the Keras Image preprocessing for image augmentation – helping the model to generalize better
6. Build a Benchmark model to check that the problem is 'solvable':
   a. Train the benchmark model on the entire training dataset.
   b. Evaluate the benchmark using f1 score
   c. Expected to see naïve prediction of a single class (a very low f1 score)
7. Build a full CNN model using Transfer Learning from the VGG19
   a. Train the model on the entire training dataset using K-Fold – while passing a calculated class_weight to the fitting process (*def get_train_class_weights(…)*)
   b. Evaluate the benchmark using f1 score
   c. Hyper-parameters tuning was first done manually based on dozens of experiments (later I added also a Keras implementation of sklearn's GridSearchCV but luckily the best hyperparameters tuning were already the ones I selected)
   d. CNN tuning - Loss function and optimizer tuning
   e. Train the CNN on full data and follow up on the learning curve
   f. The last 3 sections **7.c. 7.d. 7.e** - required many GPU hours took most of the time (hours became days)
8. Use the trained model to predict on the unseen test dataset (21128 images)
9. Present the results as described in the Evaluation metrics section

Refinement

In this section I mostly refer to the evolution of the full CNN that I was building to solve the problem.

Initial solution and experiments

The initial CNN was based on a **Transfer Learning** from a VGG19 (15) CNN pretrained on 'imagenet' .

The original first 6 VGG19 layers were frozen (not trained)

I added two Dense layers in size 64, and output layer of size 1 as I was trying to perform a binary classification 0= IDC(-) and 1=IDC(+).

I tried using a loss function of 'binary_crossentropy' with 'sigmoid' activation on the output layer – for a reason I could not entirely understand, my custom f1_score implementations didn't work well on binary classification.

At some point I changed the model to predict **a multiclass classification of two classes**, and then the f1 started to give reasonable results.

Tried GridSearchCV in Keras, which can be seen in the notebook *do_grid_search(…)* and also a "manual Grid Search" to tune the hyper parameters (final selected values in **bold**):

- batch_size = [25,**50**,100]
- optimizers = ['adam','rmsprop',**'SGD with lr=0.0001,momentum=0.2**','Adadelta']
- epochs = [5 , 10 , **15**]
- folds = [5,**7,** 10]

Final solution

I did the following changes on the CNN architecture:

Changed the model to work with multiclass classification of two classes, the loss function to 'categorical_crossentropy' and the activation function 'softmax' of the output layer.

Added BatchNormalizatioin, L2 Regularization(0.01) + Dropout (0.5)

- batch_size = 50
- optimizers = SGD with lr=0.0001 momentum=0.2
- epochs = 15
- folds = 7

During the process I changed and enhanced the vitalizations many times, until I got clear results, including the training learning curves, the classification report and the confusion matrix.

IV. Results

Model Evaluation and Validation

The final selected model is CNN which was based on VGG19 pretrained on 'imagenet' (Transfer Learning was used).

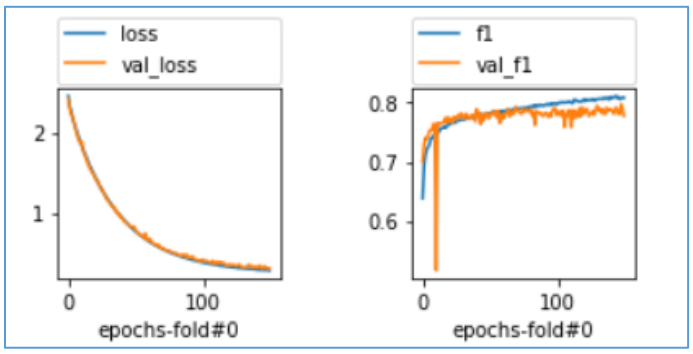
It was chosen because this approach was proven to be very good at image classification problems as explained in the 'Algorithms and Techniques' section.

During training the image data was augmented and shuffled to reduce the risk of overfitting, and in addition a few experiments were done on lower numbers of images (50K, 100K,150K – all showing lower f1 scores than when using the full dataset of ~250K)

The model generalized quite well and seemed to be quite robust, but mainly for classifying IDC(-), I assume it can be used as a helping tool medical screening processes, at least for initial screening of IDC(-) which scored over 91% accuracy – and probably can even get better with more some more tuning and additional training 'effort'.

I was not able to get such good results for IDC(+) images, possibly because I need more data and maybe slower learning rates.

The longest training I run was 150 epochs X 3 folds with an SGD optimizer and a very small learning rate, and loss was improving wel las well as the f1, and it tool 15 hours to complete:

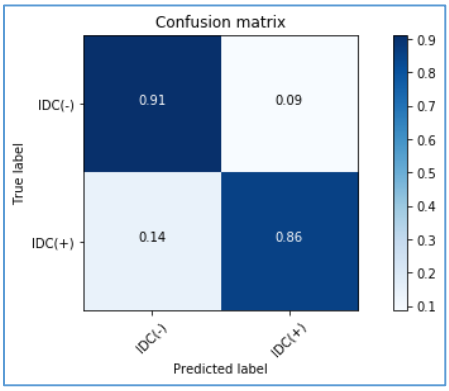


The model evaluation was performed in two independent methods:

1. As part of the **training process** using KFold cross validation- at the end of every fitting cycle, an evaluation was done on the selected 'fold' – and the validation f1_score is measured
2. At the end of the **training** (after all folds were completed) there is an additional evaluation on unseen data that was kept aside - and the validation f1_score is measured again, in addition to a full classification report and confusion matrix to describe the performance of a classification model against the ground truth (true classes)

Final classification report and confusion matrix on test dataset:

```
*** Classification Report ***
precision    recall  f1-score   support

IDC(-)       0.93      0.90      0.91    213519
IDC(+)       0.76      0.84      0.80     85133

avg / total       0.88      0.88      0.88    298652
```



Overall, I think that the results can be trusted, surely as a quick helping screening tool – possibly it can be used by the medical teams to pay more attention to samples that are identified as IDC(+).

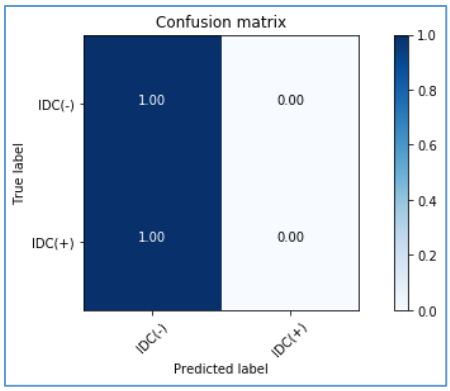
Some more ideas are presented in the following '[Improvement'](#) section

Justification

The following section compares the results of the benchmark and the final models, it shows the classification report of naïve benchmark model, which was able to classify very poorly, versus the performance of the tuned deep learning neural network (CNN) that is capable of classifying quite well - whether a given mammography image patch is 'IDC(-)' or 'IDC(+)'

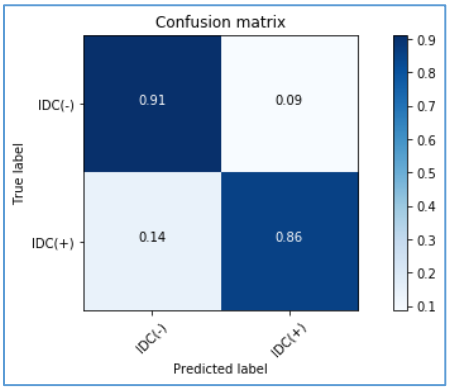
Benchmark results

```
*** Classification Report ***
             precision    recall  f1-score   support

     IDC(-)       0.70      1.00      0.82     14781
     IDC(+)       0.00      0.00      0.00      6347

avg / total       0.49      0.70      0.58     21128
```



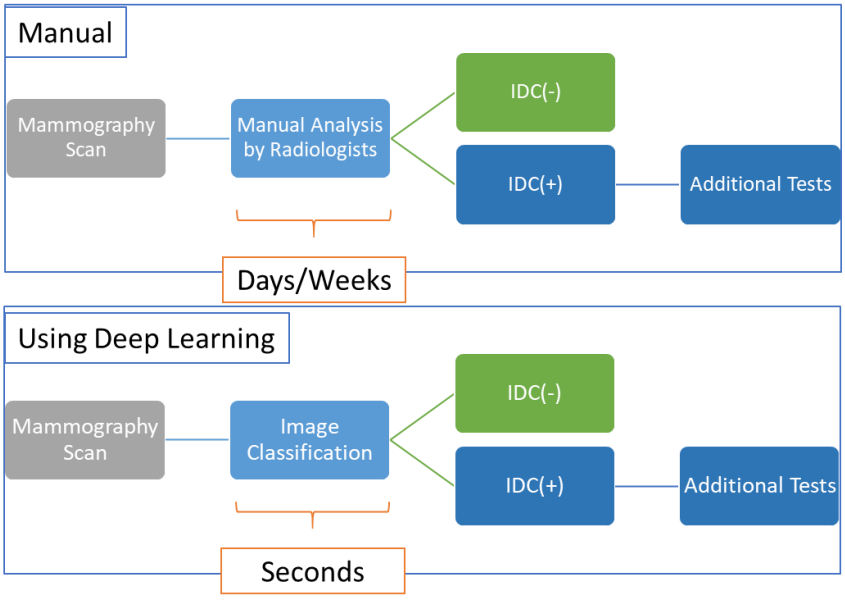
Final Model results

```
*** Classification Report ***
precision    recall  f1-score   support

IDC(-)       0.93      0.90      0.91    213519
IDC(+)       0.76      0.84      0.80     85133

avg / total       0.88      0.88      0.88    298652
```



V. Conclusion

Free Form Visualization



This project was dealing with **saving lives** – by using today's state of the art Deep Learning technologies for image classification. Hopefully it suggests a way of automating a monotonous manual process practiced today by trained medical personnel

The project suggests how to address the critical early detection of IDC by making the classification of IDC an automatic simple process, it aims to help medical teams to perform initial screening quicker and in higher quality.

Reflection

The solution presented in this paper describes how Deep Learning CNN can be used for medical images classification.

Reading about the domain of breast cancer, and IDC specifically, clarified to me just how common the phenomenon is among women, and how the technology can help improving the manual screening process that is still very common today.

I enjoyed the data exploration part, in which I found some outliers just by looking at the images, and also realized that an 'untrained eye' cannot perform this classification task.

Implementing the Transfer Learning technique, design the CNN architecture and visualizing the results was fun, during the training I could follow the learning curves and see if the model is improving.

It was quite challenging to get good f1 score (> 80% f1 score); it took many training hours (on GPU), working on smaller amounts of data didn't show great results.

Understanding the class weights imbalance and its effect on the model training, was a new thing I learned during this project.

Overall the final model and solution fits my expectations for the problem, I think this type of solution can be used in order to solve image classification problems in the medical field.

Improvement

Here are some ideas for improvement I could make:

- I could use additional image recognition techniques and mark a frame on every image of IDC (+) indicating the infected area – that would help the medical team understand **why** an image was classified as IDC(+) - I think this is something which would really help trust the model.
- Base on the existing input dataset I think that the results are quite good, however taking the medical screening very seriously, as human lives depend on it, I would make sure the model is being retrained every some time, to make sure it is still robust and generalizes well – for example changes in the image scanning machinery, may cause different images which will cause bad performance.
- In addition, I would like to run **many more epochs (meaning many more training hours)** and see if I can get better performance scores and maybe also try again with different architectures, but the 88% average seems to be quite promising (with just 15-25 epochs).
- I could keep experimenting with GridSearch for the hyperparameters tuning – the problem is that using it in practice takes a very long time (hours to days).

Thank you for reading this paper. raising cancer awareness can save lives, this work was important for me personally, even if it just 'scratched the domain surface', and I see it as yet another small step in humanity's fight against cancer.

References

1. https://en.wikipedia.org/wiki/Mammography
2. https://www.breastcancer.org/symptoms/understand_bc/statistics
3. https://www.cancerresearchuk.org/about-cancer/cancer-symptoms/why-is-early-diagnosis-important
4. https://www.hopkinsmedicine.org/breast_center/breast_cancers_other_conditions/invasive_ductal_carcinoma.html
5. https://www.nature.com/articles/s41598-018-22437-z
6. https://www.cancer.org/latest-news/if-youre-called-back-after-a-mammogram.html
7. https://www.cancer.org/cancer/breast-cancer/screening-tests-and-early-detection/mammograms/limitations-of-mammograms.html
8. https://www.uchealth.org/today/2015/07/06/how-accurate-are-mammograms
9. https://ww5.komen.org/BreastCancer/AccuracyofMammograms.html
10. https://en.wikipedia.org/wiki/F1_score
11. http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures
12. https://www.kaggle.com/paultimothymooney/breast-histopathology-images
13. https://keras.io
14. https://en.wikipedia.org/wiki/Transfer_learning
15. https://keras.io/applications/#vgg19
16. https://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/
17. http://opencv.org/
18. https://keras.io/preprocessing/image/
19. https://www.nvidia.com/en-us/data-center/dgx-1/
20. https://en.wikipedia.org/wiki/Accuracy_paradox