Project Report

Anomaly Detection with Deep Learning

Breast Cancer Classification

Prepared by:

Zeyneb Chiha

2nd Telecommunication Engineering student

Thanks Page…

I would like to thank my Professor Mr.Mahdi Louati for giving me a such opportunity to work on this project which I consider a dream as “an engineer with spirit of a doctor”,thank you for believing in me and supporting me during this period.Wish you all the best .

Finally, we would like to thank the members of the jury who, by their expertise, contribute to the promotion of this work and any person who contributed directly or indirectly to the preparation of this report.

Sincerely.

SUBJECTS LIST

FIGURES LIST

CHAPTER 1

INTRODUCTION

I)Background

1-Problematic:

Cancer has become a deadly disease and more people are suffering from Cancer and a survey says one in every 30 women suffer from this disease in their lifetime and so basically the project was first thought of because of the increase in cases of breast cancer and one thing which is very important is that if we can detect the Cancer at an early stage then there is an increased chances of it getting cured.

Breast cancer is the most common form of cancer in women, and Invasive Ductal Carcinoma (IDC) is the most common form of invasive breast cancer .Almost 80% of diagnosed breast cancers are of this subtype.

2-solution:

This project lays a foundation in making the detection of the cancer automated so that more and more people can get it diagnosed early so as get cured.

II)Project Objective

The goal of this project is to create a web based tool that can batch analyze histopathology image patches and predict if breast cancer is present. The web app will help pathologist to effectively detecting the malignant tissue .

III) Challenge

Build an algorithm to automatically identify whether a patient is suffering from breast cancer or not by looking at biopsy images. The algorithm had to be extremely accurate because lives of people is at stake.

IV)Medical studies:

Histopathology

This involves examining glass tissue slides under a microscope to see if disease is present. In this case that would be examining tissue samples from lymph nodes in order to detect breast cancer.

Lymph Node

This is a small bean shaped structure that's part of the body's immune system. Lymph nodes filter substances that travel through the lymphatic fluid. They contain lymphocytes (white blood cells) that help the body fight infection and disease.

Whole Slide Image (WSI)

A digitized high resolution image of a glass slide taken with a scanner. The images can be several gigabytes in size. Therefore, to allow them to be used in machine learning these digital images are cut up into Patches.

Patch

A patch is a small, usually rectangular, piece of an image. For example, a 50x50 patch is a square patch containing 2500 pixels, taken from a larger image of size say 1000x1000 pixels.

Sentinel Lymph Node

A blue dye and/or radioactive tracer is injected near the tumor. The first lymph node reached by this injected substance is called the sentinel lymph node. The images that we will be using are all of tissue samples taken from sentinel lymph nodes.

Invasive Ductal Carcinoma

This is the most common subtype of all breast cancers. Almost 80% of diagnosed breast cancers are of this subtype. Ductal carcinoma starts to develop in the milk ducts whereas lobular carcinoma has its origin in the lobules. Invasive carcinoma is able to leave its initial tissue compartment and can form metastases via lymphatics.

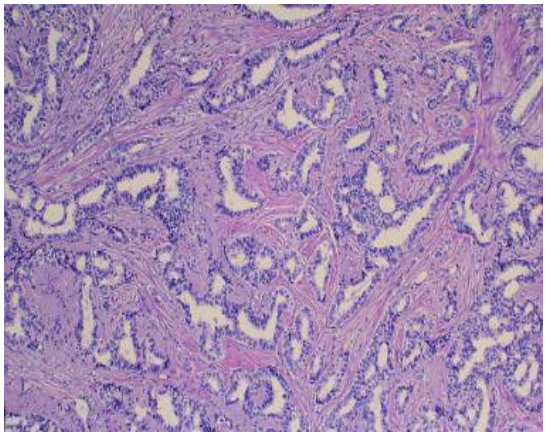


Fig-1 :Histologic characteristics

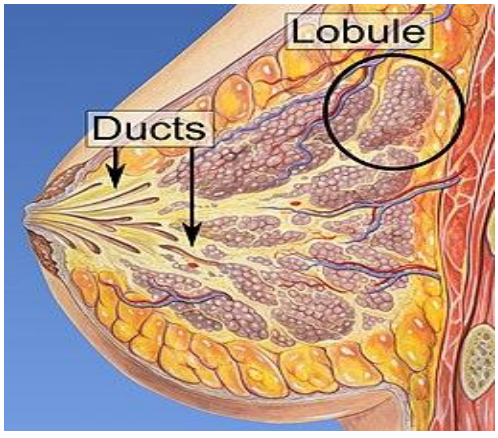


Fig-2 :Anatomy of healthy

V)WORKSPACE ENVIRONMENT

ANACONDA:

Anaconda is a free and open-source[6] distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

VISUAL CODE:

Visual Studio Code (VSC par la suite) est un éditeur de **code** open-source, gratuit et multi-plateforme (Windows, Mac et Linux), développé par Microsoft, à ne pas confondre avec **Visual Studio**, l'IDE propriétaire de Microsoft

Google Colab :

Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

CHAPTER 2

Methodology

I)INTRODUCTION

A convolution Neural Networks for breast cancer detection is a combination of many types of layers.

II)BLOCK DIAGRAM:

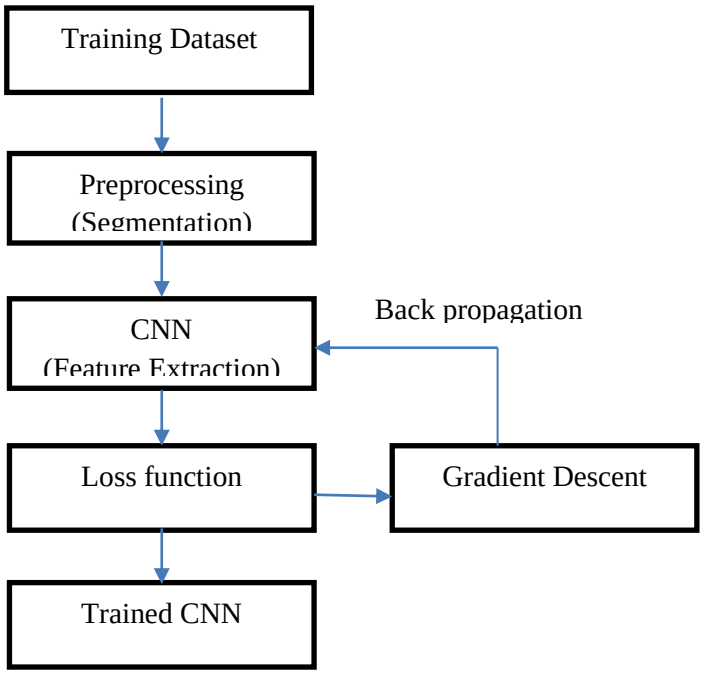


Fig 3: Training

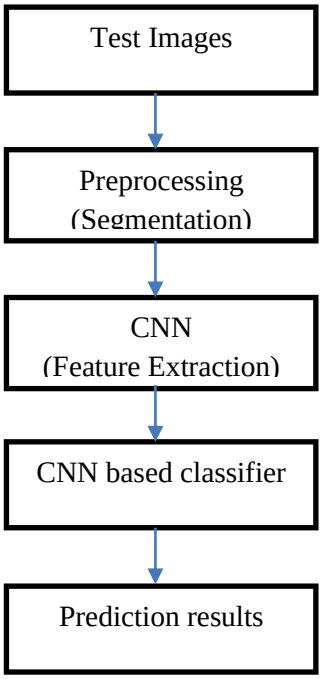


Fig 4: Testing

III)Dataset

The Breast Cancer Histopathological Image Classification (BreakHis) is composed of microscopic images of breast tumor tissue collected from 82 patients using different magnifying factors (40X, 100X, 200X, and 400X). To date, it contains 900 benign and 1051 malignant samples (700X460 pixels, 3-channel RGB, 8-bit depth in each channel, PNG format).

1-Structure

//Data

//Train

Benign

B1.png……

Malignant

M1.png….

//validation

Benign

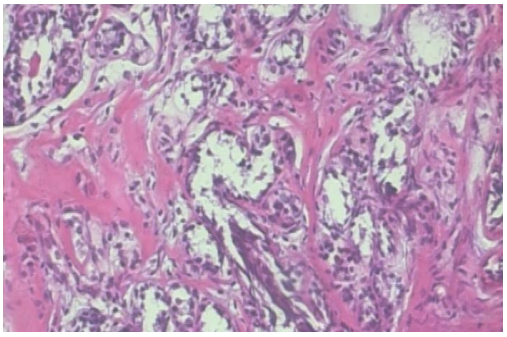
B1.png……

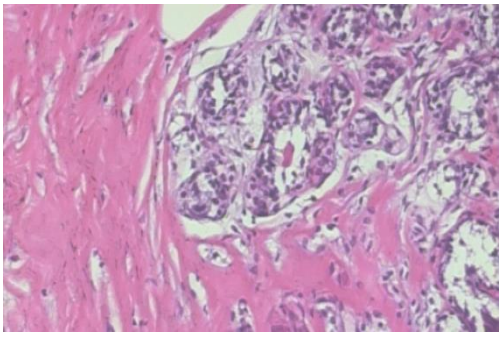
Malignant

M1.png….

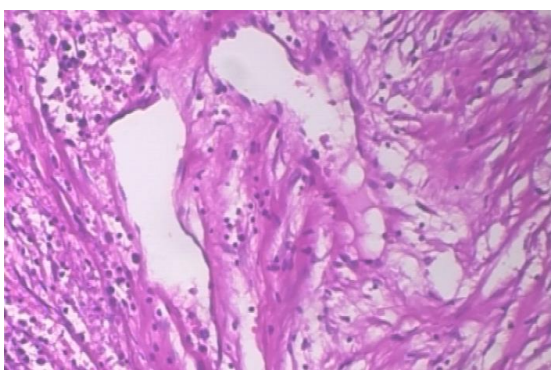
III.2-SAMPLES

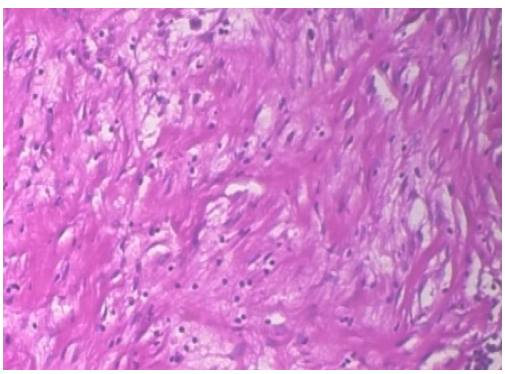
III.2.1-Benign Samples





III.2.2-Malignant samples





IV)CNN Architecture

1-Convolutional Layers: is the very first layer where we extract features from the images in our datasets.

2-Pooling Layer:When constructing CNNs, it is common to insert pooling layers after each convolution layer to reduce the spatial size of the representation to reduce the parameter counts which reduces the computational complexity.

3-A Set of Fully Connected Layers: is our RegularNet where each parameter is linked to one another to determine the true relation and effect of each parameter on the labels.

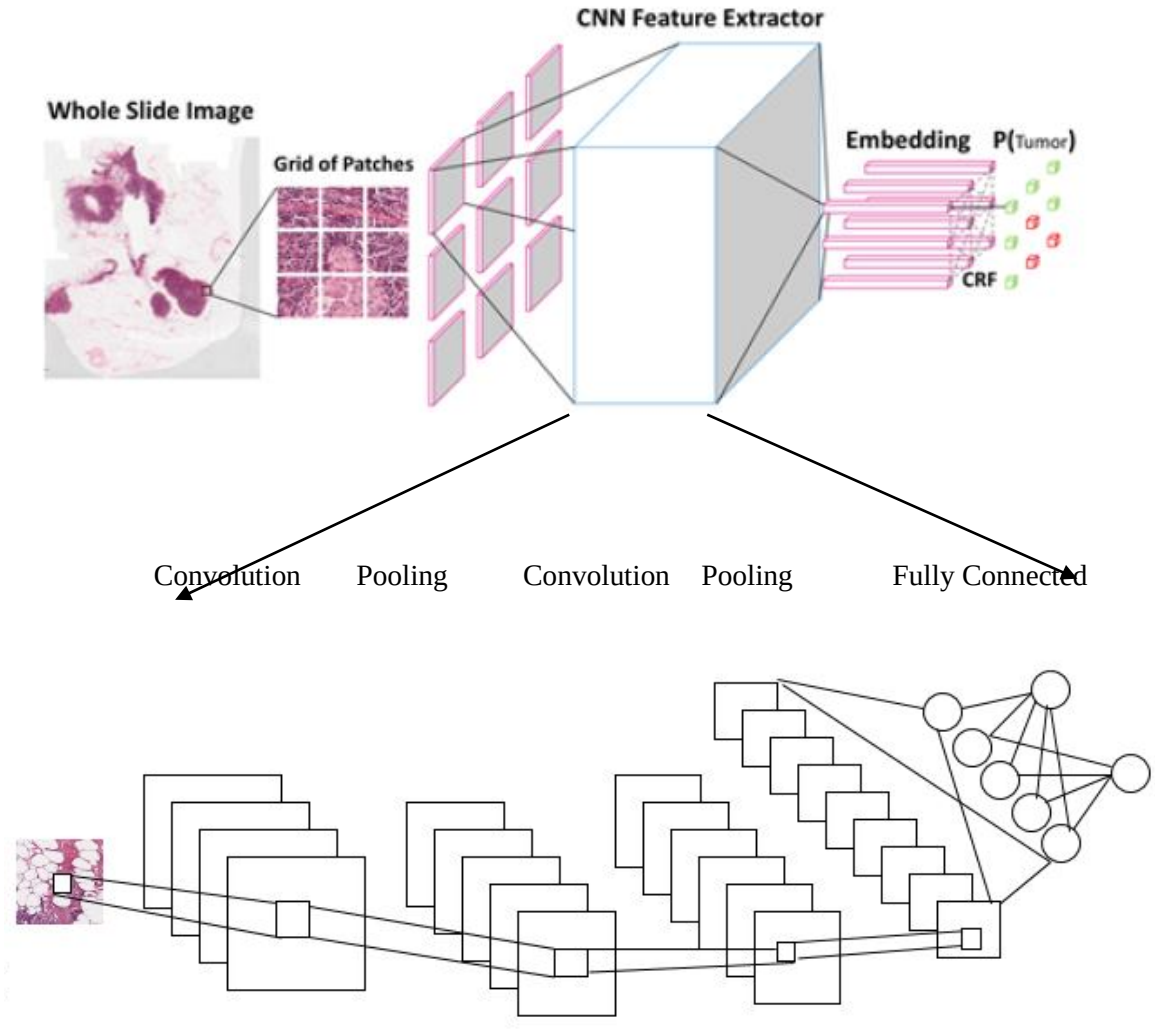


Fig 5:CNN Architecture

V)Image Classification Process

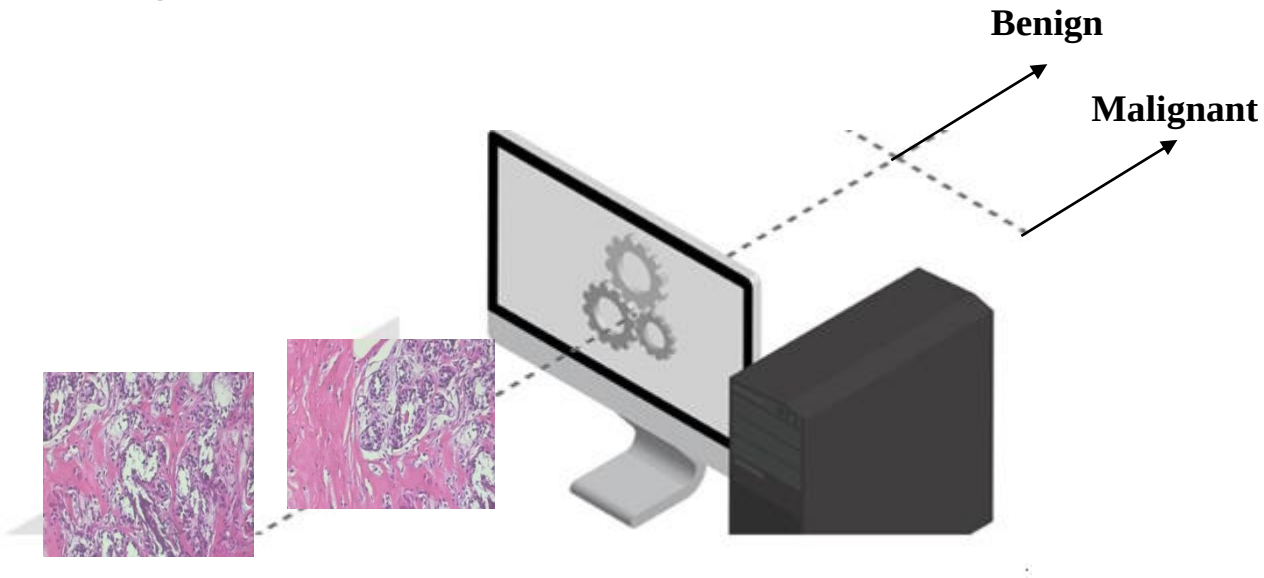


Fig.6:Image Classification Process

The complete image classification pipeline can be formalized as follows:

- Our input is a training dataset that consists of N images, each labeled with one of 2 different classes.

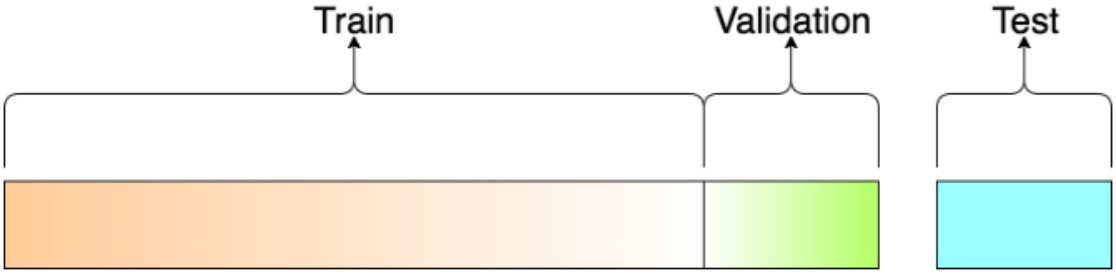
1-Data Pre-processing

- load the images in the respective folders.
- Create a numpy array of zeroes for labeling benign images and similarly a numpy array of ones for labeling malignant images.
- shuffle the dataset and converte the labels into categorical format.

  -Data splitting:

  The Test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained(using the train and validation sets). The test set is generally what is used to evaluate competing models.
- split the data-set into two sets train and test sets with 80% and 20% images respectively.

Fig.7:splits visualization



2-Data Augmentation

Data augmentation is an effective way to increase the size of the training set. Augmenting the training examples allow the network to see more diversified, but still representative data points during training.

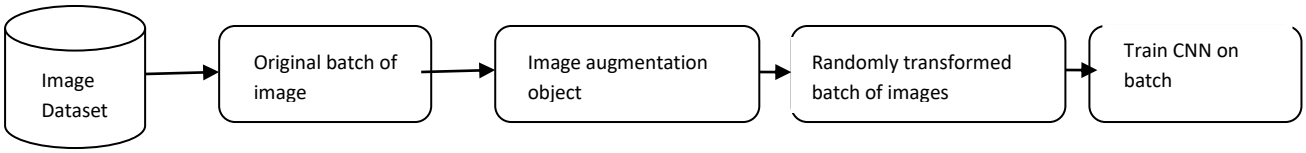


Fig.8:Keras Data Generator and Data Augmentation

- Then, we use this training set to train a classifier to learn what ever one of the classes looks like.

3-Model Building

Experinces :

I trained 4 models to get the final result :

Model1:
Pre trained Model:DenseNet201
Epochs=20
Accuracy=97%

Model2:
Pre trained Model:DenseNet201
Epochs=30
Accuracy=98,71%

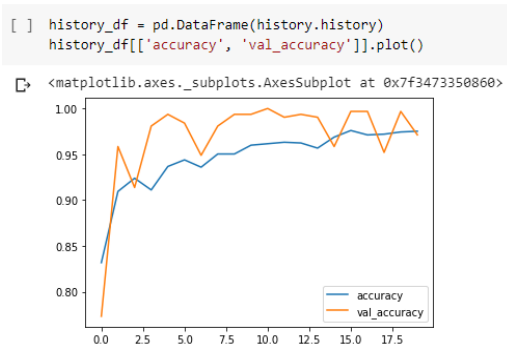


Fig.9:Model1 acc

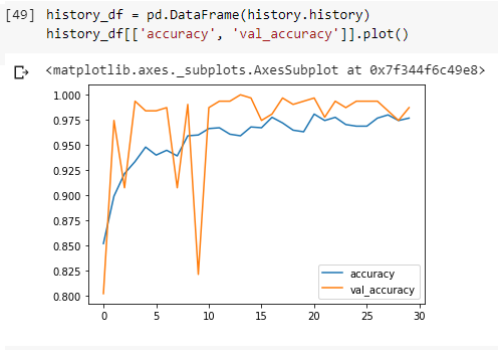
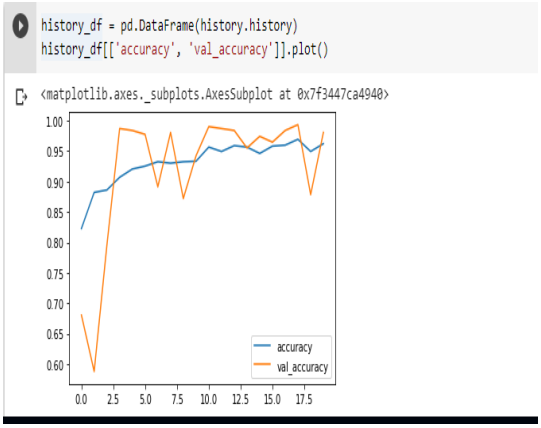


Fig.10:Model2 acc

Model3:
Pre trained Model:Resnet50
Epochs=20
Accuracy=98,08%

Model4:
Pre trained Model:ResNet50
Epochs=50
Accuracy=95,21%



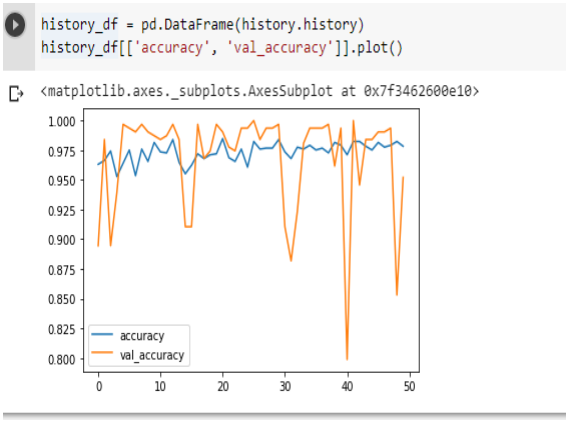


Fig.11:Mode3 acc

Fig.12: Model4 acc

Interpretation:
Throught the accuracy results, confusion matrix and other metrics we can concluse that the model 3 is the best for our dataset.

- ResNet-50 as the pre trained weights which is already trained in the Imagenet competition.
  ResNet is one of the most powerful deep neural networks which has achieved fantabulous performance results in the ***ILSVRC 2015*** classification challenge

  -Architecture of ResNet-50

  Has ***4 stages*** as shown in the diagram below**.** The network can take the input image having height, width as multiples of 32 and 3 as channel width.
  Every ResNet architecture performs the initial convolution and max-pooling using 7×7 and 3×3 kernel sizes respectively.

  Stage 1 of the network starts and it has 3 Residual blocks containing 3 layers each. The size of kernels used to perform the convolution operation in all 3 layers of the block of stage 1 are 64, 64 and 128 respectively. The curved arrows refer to the identity connection.

  The dashed connected arrow represents that the convolution operation in the Residual Block is performed with stride 2, hence, the size of input will be reduced to half in terms of height and width but the channel width will be doubled.
  As progress from one stage to another, the channel width is doubled and the size of the input is reduced to half.

Finally, the network has an Average Pooling layer followed by a fully connected layer having 1000 neurons (ImageNet class output).

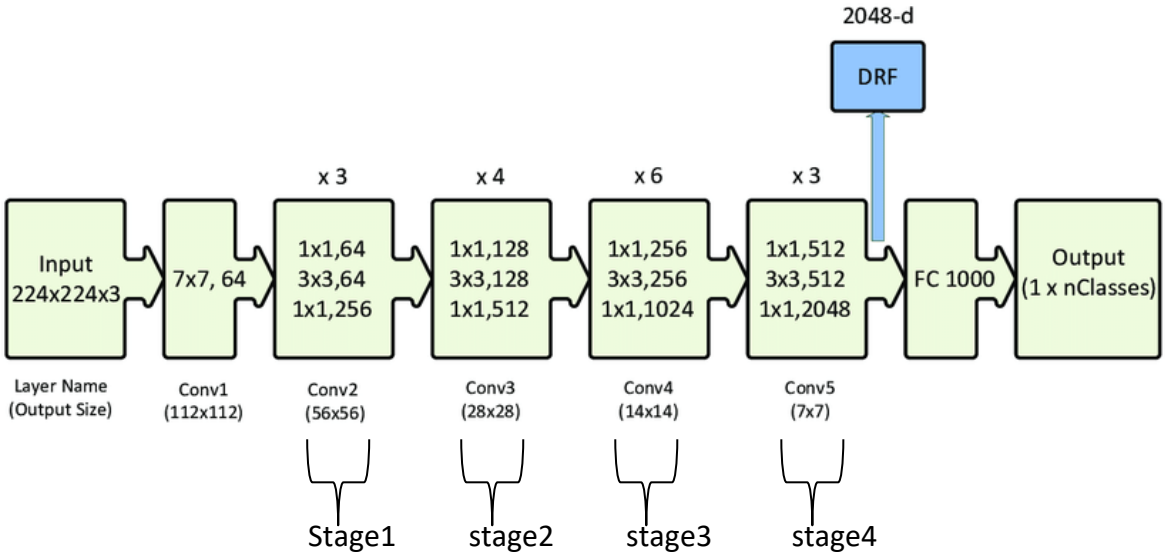


Fig.13: Architecture diagram of ResNet50

- Globalaveragepooling layer followed by 50% dropouts to reduce over-fitting.
- Batch normalization and a dense layer with 2 neurons for 2 output classes benign and malignant with softmax as the activation function.
- Adam as the optimizer and binar y-cross-entropy as the loss function.
- Batch size: using smaller batch sizes have been shown to have faster convergence to good results.

3-4-Model summary:

```
94658560/94653016 [==============================] - 1s 0us/step
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, None, None, 2048)  23587712
_________________________________________________________________
global_average_pooling2d_7 ( (None, 2048)              0
_________________________________________________________________
dropout_4 (Dropout)          (None, 2048)              0
_________________________________________________________________
batch_normalization_4 (Batch (None, 2048)              8192
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 4098
=================================================================
Total params: 23,600,002
Trainable params: 23,542,786
Non-trainable params: 57,216
_________________________________________________________________
```

Fig.14:Model summary

4-Model Training

- ModelCheckpoint:a callback that save a copy of the best performing model only when an epoch that improves the metrics ends.
- ReduceLROnPlateau: a callback that monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.
- Epochs=20:One **epoch** is when an entire dataset is passed both forward and backward through the neural network only once

```
Epoch 1/20
78/78 [==============================] - 219s 3s/step - loss: 0.3959 - accuracy: 0.8317 - val_loss: 0.5140 - val_accuracy: 0.7732
Epoch 2/20
/usr/local/lib/python3.6/dist-packages/keras/callbacks/callbacks.py:1042: RuntimeWarning:

Reduce LR on plateau conditioned on metric `val_acc` which is not available. Available metrics are: val_loss,val_accuracy,loss,accuracy

/usr/local/lib/python3.6/dist-packages/keras/callbacks/callbacks.py:707: RuntimeWarning:

Can save best model only with val_acc available, skipping.

78/78 [==============================] - 113s 1s/step - loss: 0.2254 - accuracy: 0.9095 - val_loss: 0.1193 - val_accuracy: 0.9585
Epoch 3/20
78/78 [==============================] - 113s 1s/step - loss: 0.2037 - accuracy: 0.9239 - val_loss: 0.2025 - val_accuracy: 0.9137
Epoch 4/20
78/78 [==============================] - 113s 1s/step - loss: 0.2056 - accuracy: 0.9111 - val_loss: 0.0542 - val_accuracy: 0.9808
Epoch 5/20
78/78 [==============================] - 113s 1s/step - loss: 0.1551 - accuracy: 0.9367 - val_loss: 0.0183 - val_accuracy: 0.9936
Epoch 6/20
78/78 [==============================] - 113s 1s/step - loss: 0.1568 - accuracy: 0.9439 - val_loss: 0.0294 - val_accuracy: 0.9840
Epoch 7/20
78/78 [==============================] - 113s 1s/step - loss: 0.1778 - accuracy: 0.9359 - val_loss: 0.0925 - val_accuracy: 0.9489
Epoch 8/20
78/78 [==============================] - 113s 1s/step - loss: 0.1231 - accuracy: 0.9503 - val_loss: 0.0444 - val_accuracy: 0.9808
```

Fig.15:Model Training

CHAPTER 3

Results

VI)Performance Metrics

- **Accuracy** - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best.

$$\text{Accuracy} = TP+TN/TP+FP+FN+TN$$

In our case, accuracy is about 98%.

```
[64] # evaluate the model
     score = model.evaluate(x_val, y_val, verbose=0)
     print("%s: %.2f%%" % (model.metrics_names[1], score[1]*100))

     accuracy: 98.08%
```

- Loss vs epoch:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f347658c908>
```

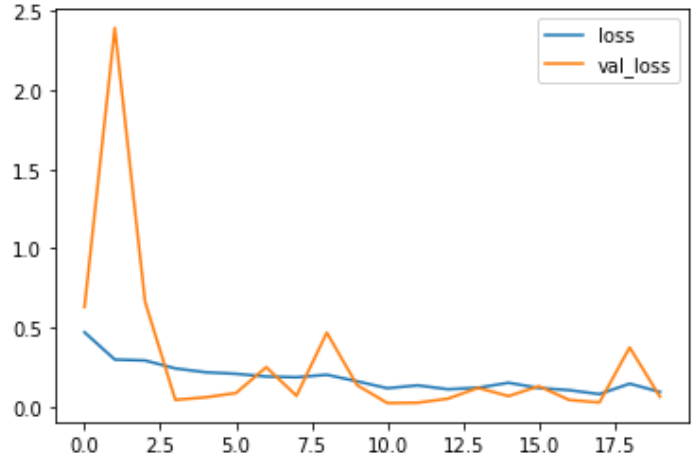


Fig.16:Model performance

- Accuracy vs epoch:

```
history_df = pd.DataFrame(history.history)
history_df[['accuracy', 'val_accuracy']].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3447ca4940>
```

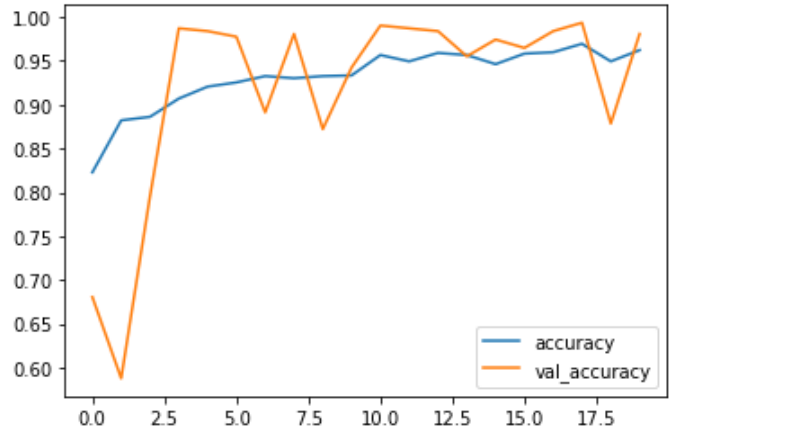


Fig.17:Model performance

- Confusion Matrix

A confusion matrix is used to describe the performance of a classification model (a "classifier") in binary data for which the true values are known as well. In its simplest and most typical presentation, it is a special contingency table with two dimensions used to evaluate the results of the test or algorithm.

Each row of the matrix represents the ascribed or attributed class while each column represents the actual condition (the truth). The cells of the confusion matrix report the number of true positives and false positives, and false negatives and true negatives.

```
Confusion matrix, without normalization
[[284   2]
 [  2 144]]
```

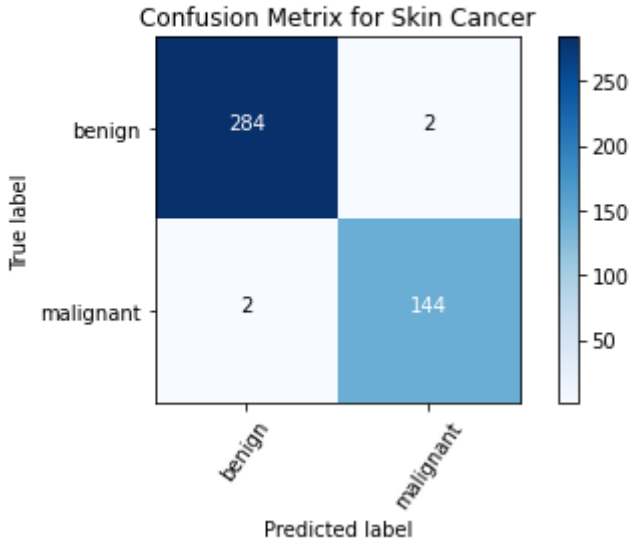


Fig.18:Confusion Matrix

- ROC Curves

The ROC curve plots the true-positive rate (TPR) against the false-positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity (or probability of detection in machine learning). The false-positive rate is calculated as 1–specificity.

```
#plt.savefig(ROC_PLOT_FILE, bbox_inches='tight')
plt.close()
```

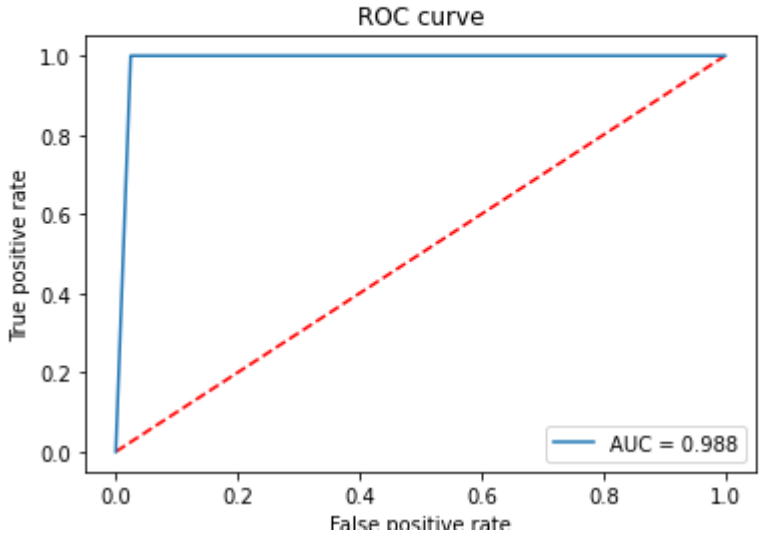


Fig.19:Roc Curves

- **Precision** - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$\text{Precision} = TP/TP+FP$$

- **Recall** (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class.

$$\text{Recall} = TP/TP+FN$$

- **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.

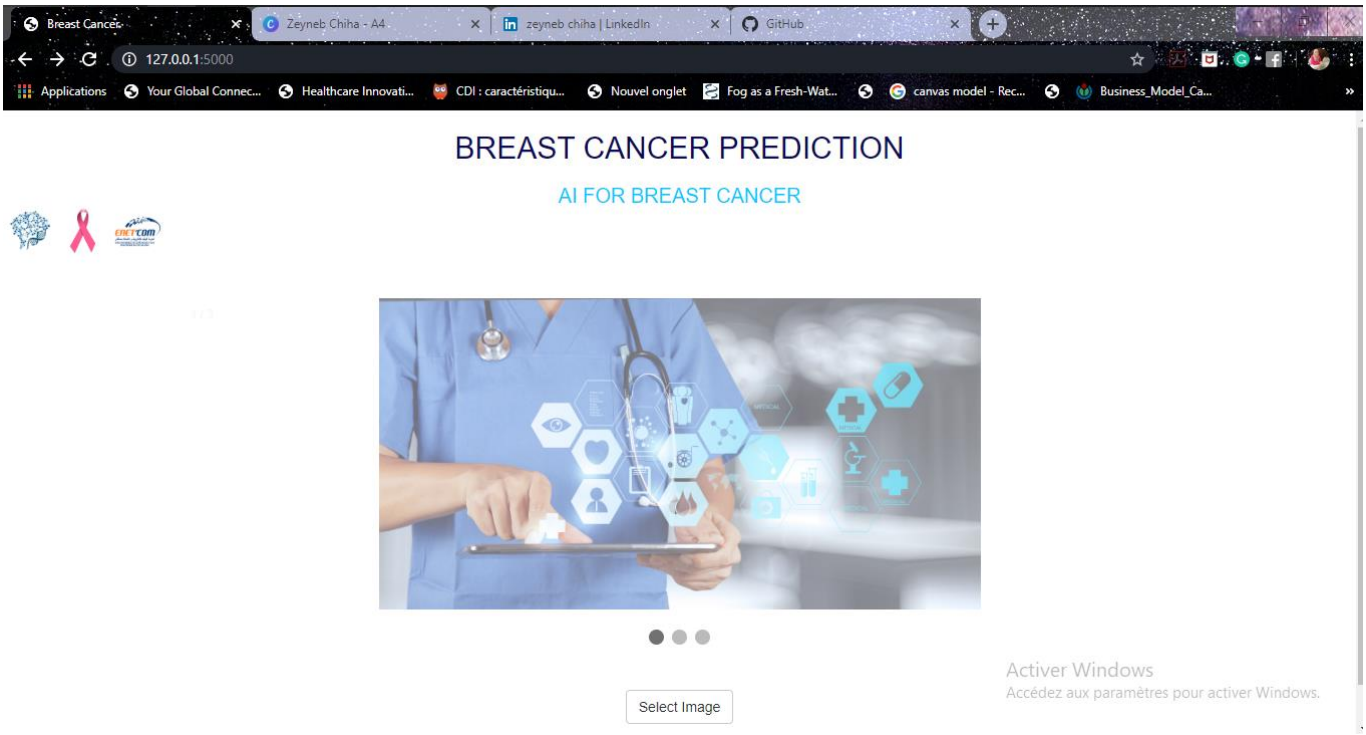
  F1 Score = 2*(Recall * Precision

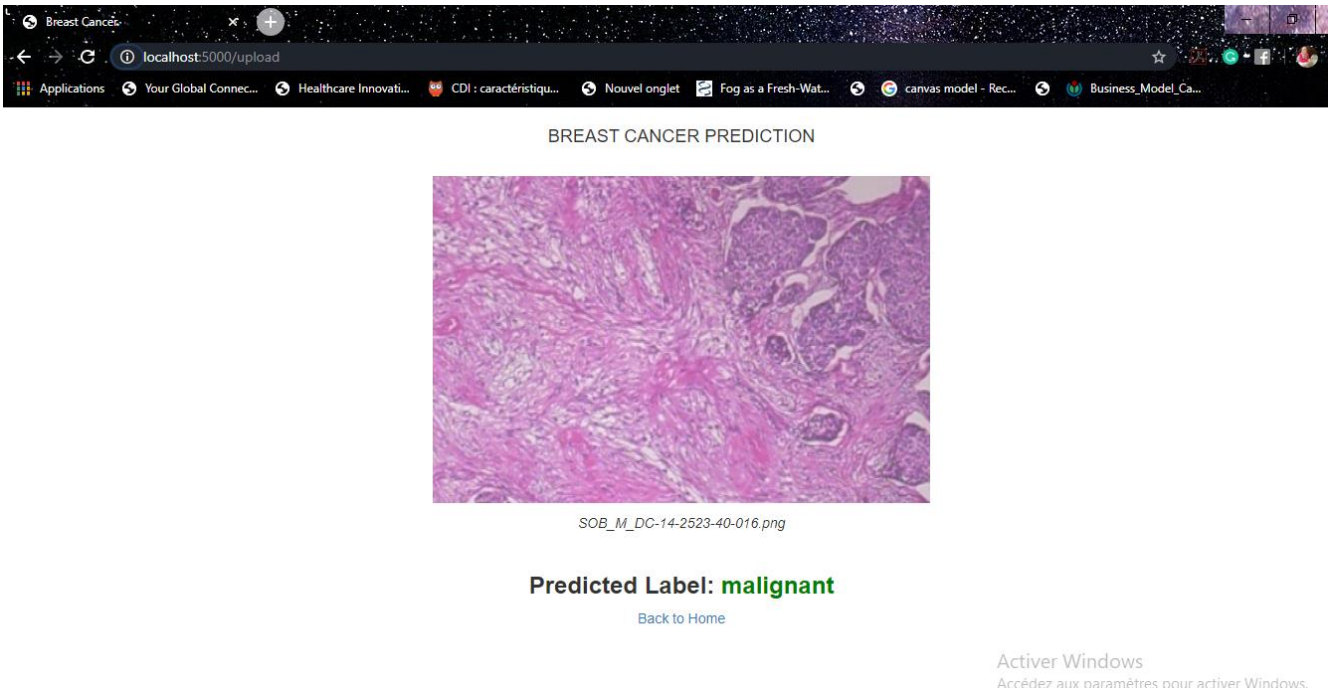
CHAPTER 4

CNN Model Deployment

Flask

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. it is one of the most popular Python web application frameworks.





Remaining tasks

- Work more on the web application
- Fix over-fitting and imbalanced-data issue.
- Improve the model's performance

Conclusion:

In this project I worked on improving the CNN model metrics performances , by training various model and image pre-processing methods.In fact can say that reached my object by

making the accuracy on 98% but I still believe through my experience that over-fitting which is caused by imbalanced data is a big challenge in Deep learning medical application and it is the common challenge with real dataset (medical data)and I will try my best to work on it more for a better medical atomization .

REFERENCES

[1Dipti Pawar :Improving Performance of Convolutional Neural Network.

[2] Abhinav Sagar: Convolutional Neural Network for Breast Cancer Classi_cation

[3] Maria Yatsenko,Market Research Specialist : How to Implement Artificial Intelligence for Solving Image Processing Tasks.

[4] **Deep learning** Yann LeCun1,2, Yoshua Bengio3 & Geoffrey Hinton4,5.

[5] **Nan Wu[b], Jason Phang[b], Jungkyu Park[b], Yiqiu Shen[b], Zhe Huang[b], Masha Zorin[h*], Stanisław Jastrz ¸ebski[i], Thibault Févry[b],Joe Katsnelson[f], Eric Kim[a], Stacey Wolfson[a], Ujas Parikh[a], Sushma Gaddam[a], Leng Leng Young Lin[a], Kara Ho[j*], Joshua D.Weinstein[a], Beatriu Reig[a,d], Yiming Gao[a,d], Hildegard Toth[a,d], Kristine Pysarenko[a,d], Alana Lewin[a,d], Jiyon Lee[a,d], KrystalAirola[a], Eralda Mema[a], Stephanie Chung[a], Esther Hwang[a], Naziya:**
Deep Neural Networks Improve Radiologists' Performance in Breast Cancer Screening

[6] Deep Learning to Improve Breast Cancer Early Detection on Screening Mammography

[7] Adrian Rosebrock :breast cancer classification with keras and deep learning

[8] **George Seif :** Handling Imbalanced Datasets in Deep Learning

[9] Cancer metastasis detection with neural conditional random field (NCRF)