BREAST CANCER DETECTION USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

SUBHASH KUMAR YADAV (196301097)
ARYAN JAISWAL (196301014)
AYUSH PANDEY (1963010)

in partial fulfilment for the award
of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING & TECHNOLOGY

GURUKULA KANGRI (DEEMED TO BE) UNIVERSITY

BONAFIDE CERTIFICATE

This to certify that this project report entitled “BREAST CANCER DETECTION USING MACHINE LEARNING” is the bonafide work of “SUBHASH KUMAR YADAV(196301097), ARYAN JAISWAL (196301014), AYUSH PANDEY (196301024)” submitted in partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science & Engineering of the Faculty of Engineering & Technology, Gurukula Kangri (Deemed to be) University Haridwar, during the academic year 2019-2023, is a bonafide record of work carried out under my guidance and supervision.

Signature

DR. MAYANK AGARWAL

(HEAD OF THE DEPARTMENT)

COMPUTER SCIENCE AND ENGINEERING

FACUTLTY OF ENGINEERING AND TECHNOLOGY

GURUKULA KANGRI (DEEMED TO BE) UNIVERSITY

HARIDWAR, PIN-249404 (U.K.)

Signature

MR. Deepak Painuali

(MENTOR)

COMPUTER SCIENCE AND ENGINEERING

FACUTLTY OF ENGINEERING AND TECHNOLOGY

GURUKULA KANGRI (DEEMED TO BE) UNIVERSITY

HARIDWAR, PIN-249404 (U.K.)

DISCLAIMER

This project, Detection of Breast Cancer using Machine Learning and is completely functional project has been prepared by the author under the Minor Project of the Faculty of Engineering & Technology, Gurukula Kangri (Deemed to be) University, for academic purposes only. The views expressed in the report are personal to the students and do not necessarily reflect the view of the FET, GKV or any of its staff or personnel and do not bind the Faculty of Engineering & Technology, Gurukula Kangri Vishwavidyalaya in any manner. This report and functional project are the intellectual property of the Faculty of Engineering & Technology, Gurukula Kangri Vishwavidyalaya and the same or any part there of may not be used in any manner whatsoever, without express permission of the Faculty of Engineering & Technology, Gurukula Kangri Vishwavidyalaya, in writing.

Yours student

SUBHASH KUMAR YADAV (196301097)

ARYAN JAISWAL (196301014)

AYUSH PANDEY (1963010)

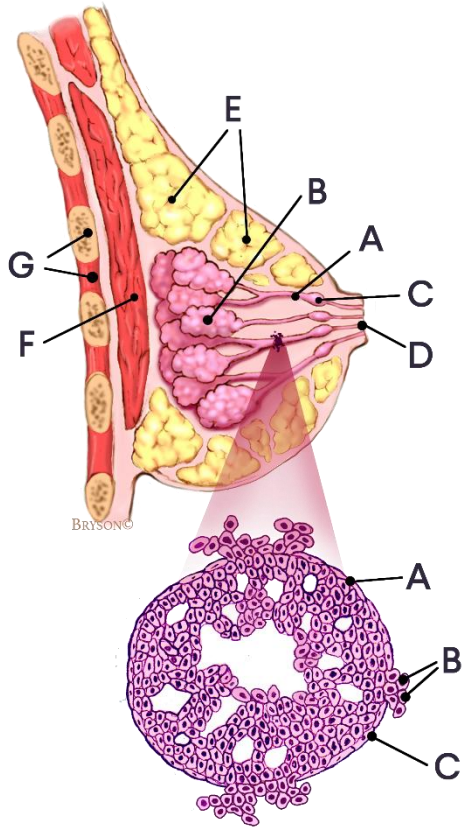
INDEX

ABSTRACT

Abstract Creating successful strategies for the robotized discovery of IDC stays a difficult issue for bosom malignant growth finding. As of late, Cruz and his colleagues proposed an AI approach for discovery of obtrusive ductal carcinoma (IDC) from entire picture slides containing bosom malignant growth cells. Their strategy, in light of a Convolutional Neural Network (CNN), does not have to handcraft highlights from pictures. Their work has the capability of reforming malignant growth identification, advancing further innovative work in this energizing heading. Propelled by Cruz's work, our group explored different CNN models for computerized recognition of bosom malignant growth. We originally executed a standard CNN like Cruz'ss, and after that all-encompassing it to four distinct models. All designs were prepared over a huge dataset of roughly 275,000, 0:43/14:53 50x50 RGB picture patches.

INTRODUCTION

Invasive ductal carcinoma (IDC), also called infiltrating ductal carcinoma, is the most common type of breast cancer. About 75% of all breast cancers are IDC, according to the American Cancer Society. Invasive means the cancer has spread into surrounding breast tissues. Ductal means the cancer started in the milk ducts, the tubes that carry milk from the lobules to the nipple. Carcinoma refers to any cancer that begins in the skin or other tissues that cover internal organs, such as breast tissue. The American Cancer Society estimates about 287,850 new cases of invasive breast cancer to be diagnosed in women in 2022, most of which are expected to be IDC.



Normal breast with invasive ductal carcinoma (IDC) in an enlarged cross-section of the duct Breast profile: A Ducts B Lobules C Dilated section of duct to hold milk D Nipple E Fat F Pectoralis major muscle

G Chest wall/rib cage Enlargement: A Normal duct cell B Ductal cancer cells breaking through the basement membrane C Basement membrane.

Signs and symptoms of invasive ductal carcinoma

In many cases, invasive ductal carcinoma causes no symptoms and is found after your doctor sees a suspicious area on a screening mammogram. In other cases, you or your doctor may feel a lump or mass in your breast. Any of the following changes in the breast can be a first sign of invasive ductal carcinoma: swelling of all or part of the breast skin irritation skin dimpling, sometimes looking like an orange peel breast or nipple pain nipple turning inward (retraction) nipple discharge, other than breast milk redness, scaliness, or thickening of the nipple or breast skin a lump or swelling in the underarm area

Subtypes and grades of invasive ductal carcinoma

There are several rare subtypes of invasive ductal carcinoma. These are often named for features seen when the cells are looked at under a microscope, such as the way the cells are arranged. The symptoms, diagnosis, staging, treatment options, and survivorship care are generally the same for all IDC subtypes. Still, the cancer grade can vary from low to high, depending on the IDC subtype. A cancer's grade is based on how much cancer cells look like normal cells. Cancer grade is different from cancer stage.

There are three cancer grades:

Grade 1, or low grade,usually refers to cancer that grows slowly and is less likely to spread.

Grade 2, or moderate grade, refers to cancer that is growing faster than a grade 1 but slower than a grade 3.

Grade 3, or high grade, refers to cancer that is growing faster than grade 1 and grade 2 and which is more likely to spread.

Typically, cancer that has a low grade has better treatment outcomes (prognosis). Learn more about the following IDC subtypes below:

- ❖ Tubular carcinoma of the breast
- ❖ Medullary carcinoma of the breast
- ❖ Mucinous carcinoma of the breast
- ❖ Papillary carcinoma of the breast
- ❖ Cribriform carcinoma of the breast
- ❖ Metaplastic carcinoma of the breast

❖ Tubular carcinomas of the breast

Tubular carcinomas account for less than 2% of all breast cancers. When looked at under a microscope, the cells of a tubular carcinoma look like tubes. These tumors tend to be low-grade. Tubular carcinomas are usually hormone receptor-positive and HER2-negative.

❖ Medullary carcinomas of the breast

Medullary carcinomas account for less than 5% of all breast cancers.

It is called “medullary” carcinoma because the tumor is a soft, fleshy mass that resembles a part of the brain called the medulla. Medullary carcinomas are more common in women who have a BRCA1 mutation. Also, medullary carcinomas are often triple-negative, which means they are estrogen receptor-negative, progesterone receptor-negative, and HER2-negative. But unlike most triple-negative breast cancers, which tend to be more aggressive, medullary carcinoma doesn’t grow quickly and usually doesn’t spread outside the breast to the lymph nodes. Medullary carcinoma cells are usually high-grade in their appearance and low-grade in their behavior. Despite looking like aggressive, highly abnormal cancer cells, medullary carcinoma cells don’t act like them.

❖ Mucinous carcinomas of the breast

Mucinous carcinomas — also called colloid carcinomas — account for less than 2% of all breast cancers. In mucinous carcinoma, the tumor is made up of abnormal cells that "float" in pools of mucin, a key ingredient in mucus. Mucinous carcinomas are usually low-grade, and usually don't spread outside the breast to the lymph nodes. Mucinous carcinomas are usually hormone receptor-positive and HER2-negative.

❖ Papillary carcinomas of the breast

Papillary carcinomas of the breast account for less than 1% of all breast cancers. When the cells are looked at under a microscope, they have finger-like projections or "papules." Papillary carcinomas are usually small, hormone receptor-positive, and HER2-negative. In most cases of papillary carcinoma, ductal carcinoma in situ (DCIS) is also present. DCIS is non-invasive breast cancer that has not spread outside the milk ducts where it started.

❖ Cribriform carcinomas of the breast

Cribriform carcinomas account for less than 1% of all breast cancers. In invasive cribriform carcinoma, the cancer cells invade the stroma (connective tissues of the breast) in nestlike formations between the ducts and lobules. Within the tumor, there are distinctive holes in between the cancer cells, making it look something like Swiss cheese. Invasive cribriform carcinoma is usually low grade.

❖ Metaplastic carcinomas of the breast

Metaplastic carcinomas account for less than 1% of all breast cancers. Metaplastic breast cancers contain abnormal ductal cells, but also contain cells that look like the soft tissue and connective tissue in the breast. The ductal cells have changed their form to become completely different cells, though it's not clear how or why this happens. When cells change form it's called metaplasia, which gives this type of breast

cancer its name. Metaplastic breast cancer is considered more aggressive and is usually high grade and triple-negative, meaning it is estrogen receptor-negative, progesterone receptor-negative, and HER2-negative.

AIM & OBJECTIVES

20% of Breast Cancer can be reduced with early detection. The basic aim of this project is to develop an efficient system which is able to predict the breast cancer in early stages based on the automatic diagnosis of the breast regions. The system can also be used for the detection of the future stages of breast cancer.

Our Aim is to achieve 85% accuracy for this Machine learning alogorithm.

FLOWCHART

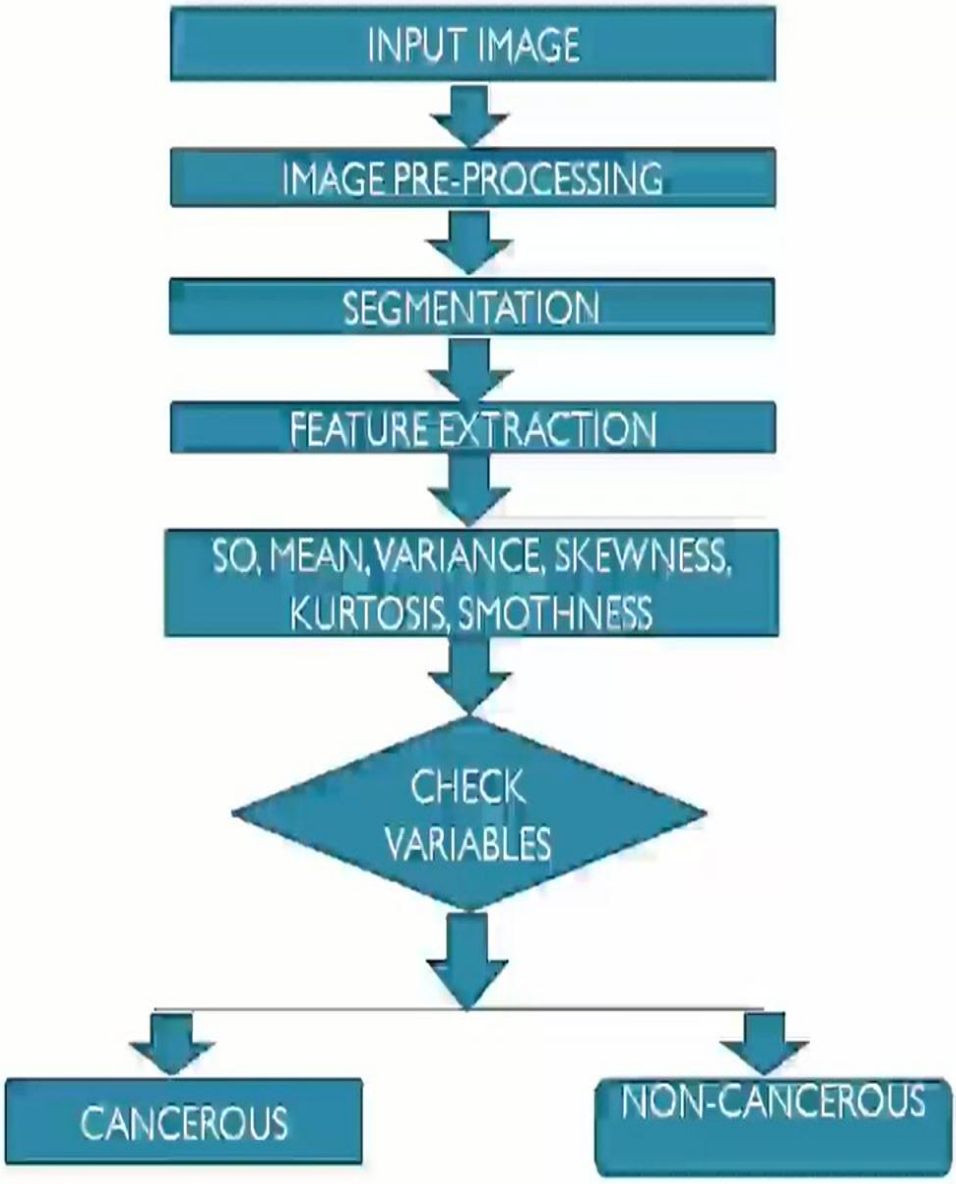

INPUT IMAGE
IMAGE PRE-PROCESSING
SEGMENTATION
FEATURE EXTRACTION
SO, MEAN, VARIANCE, SKEWNESS, KURTOSIS, SMOTHNESS
CHECK VARIABLES
CANCEROUS
NON-CANCEROUS

IMAGE PRE PROCESSING

- Image preprocessing is a way to improve the quality of image, so that the consequential image is better than the original one. Here, mean filter and median filter are presented for preprocessing of selecting the mammograms.
- Both mean and median filters are used to remove noise. This preprocessing image is used as the input for image segmentation.
- The aim of this process is an improvement of the image data that suppresses unwanted distortions or enhances some features important for further processing with CT images, they already contain lesser amount of noise

IMAGE SEGMENTATION

- Image segmentation is an essential process for most image analysis subsequent tasks. Segmentation divides an image into its constituent regions or objects. The goal of segmentation is to make simpler or change the representation of an image into something that is more meaningful and easier to analyze.
- Image segmentation is the process in which a digital image is divided into multiple segments. Image segmentation process involve operations like thresholding, edge detection, watershed transform.
- The goal of segmentation is to simplify and/or change the representation of the image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images

FEATURE EXTRACTION

This stage is an important stage that uses algorithms and techniques to detect and isolate various desired portions or shapes of a given image. When the input data to an algorithm is too large to be processed and it is suspected to be notoriously redundant, then the input data will be transformed into a reduced representation set of features

CLASSIFICATION

After the structure is analyzed, each and every region identified is evaluated individually (scoring) for the probability of a True Positive (TP). Several methods exists for the classification process. Some of them are, rule based methods, minimum distance classifier, cascade classifier, Bayesian classifier, Multilayer perception, Radial Basis Function network (RBF), Support Vector Machine (SVM), Artificial Neural Networks, Fuzzy logic,CNN etc.

METHODOLGY

The algorithm is divided into two stages: training and verification. In the training stage, the first step is to clean up the IDC row data by removing rows with all zero values. We then keep only those IDCs that have been identified via wet lab as possible biomarkers for breast cancer detection. These biomarkers are classified as clinically verified IDC's Three feature selection methods. Information Gain, Chi Squared and LASSO are applied to independently rank the importance of IDCs. The resulting feature vectors are then fed to classification algorithms. For this purpose, two classifiers, Random Forest and Support Vector Machines (SVMs), are applied to train the model. Feature selection is performed on the selected subset of the breast-cancer dataset and these IDC features are ranked by each feature selection method individually From these ranked features, different subsets were selected and were then fed into the classification algorithms.

Data Set Description

WBCD repository provided the numerical dataset. Features are composed of fine needle aspirate (FNA) of a breast mass. There are 30 features that was extracted to describe characteristics of cell nuclei present in the scanned images. The dataset consists of 569 patients ,212 have an outcome of malignancy and 357 are Benign. The classes in the dataset are separated into 2 or 4 groups, with 2 corresponding to the benign case and 4 corresponding to the malignant case.

IDC dataset consists of 162 whole mount slide images,2,77,524 patches were extracted among which 1,98,738 are non-IDC and 78,786 are IDC. They are labelled 1 and 0, where 1 indicates cells with IDC characteristics and 0 indicates cell with non-IDC characteristics.

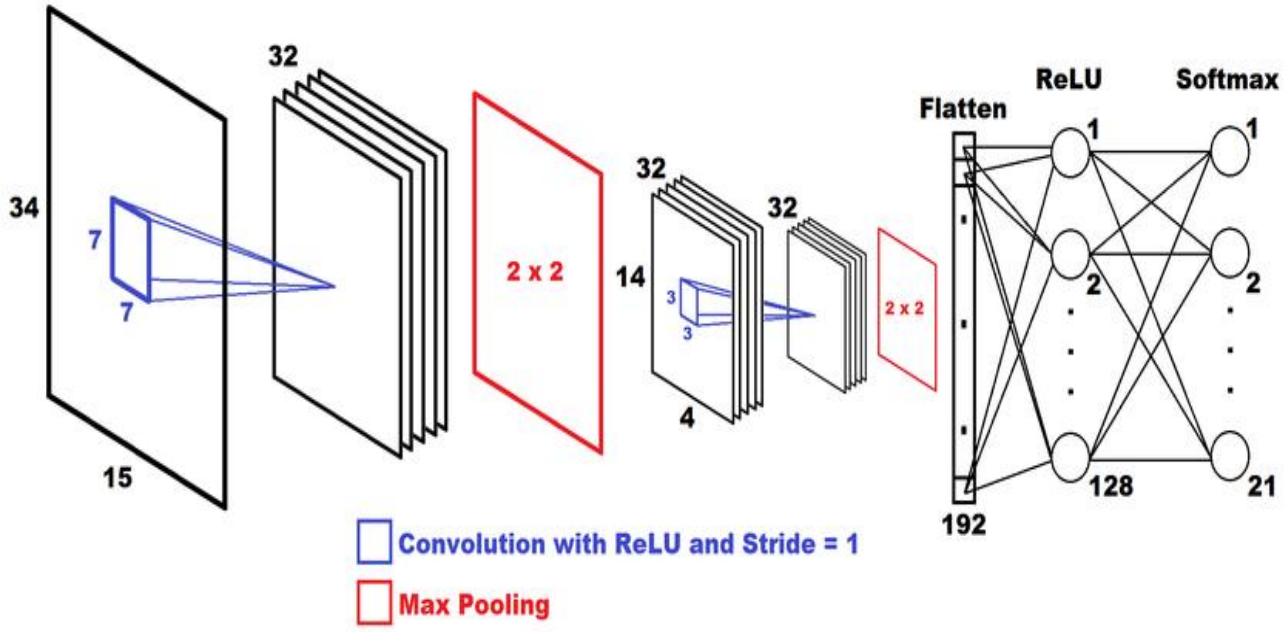
Data Pre-processing

The term "data pre-processing" refers to the process of converting unstructured data to structured data, as well as resizing and removing undesirable data from a dataset. The dataset's missing traits are replaced by the mean value. The data is then randomly selected from the dataset to ensure that the data is circulated properly.

Training and Testing Phase

This phase extracts the features from the dataset, and thetesting phase will deliver new data to be examined to see how well our algorithm works and behaves when it comes to prediction. As previously stated, the dataset is divided into two pieces. To avoid fitting, cross-validation is performed. Each iteration of our approach uses a ten-fold approach to portion data, with nine-fold used for training and the remaining for testing.

Proposed convolution neural network for image dataset analysis



- CNN's are a type of neural network that are really good at identifying the features of an image.
- And before long, the CNN is smart enough to know what features make up a certain class— or not.

  It detects an end now the next layer it goes through these edges puts together similar one to create more complex features.
- Train CNN using feed forward and soft max.
- Use pretrained model to obtain the feature representation after the final convolutional layer.

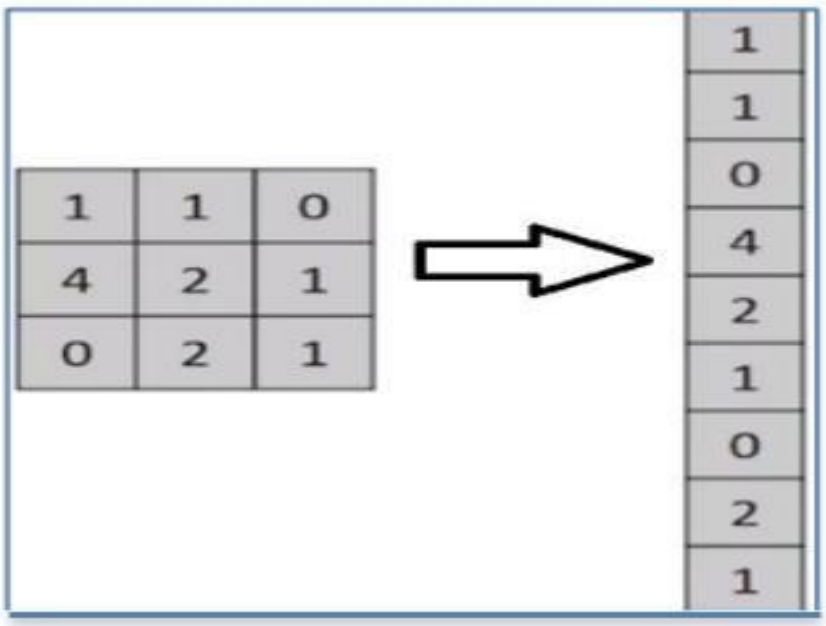
Train and test the model

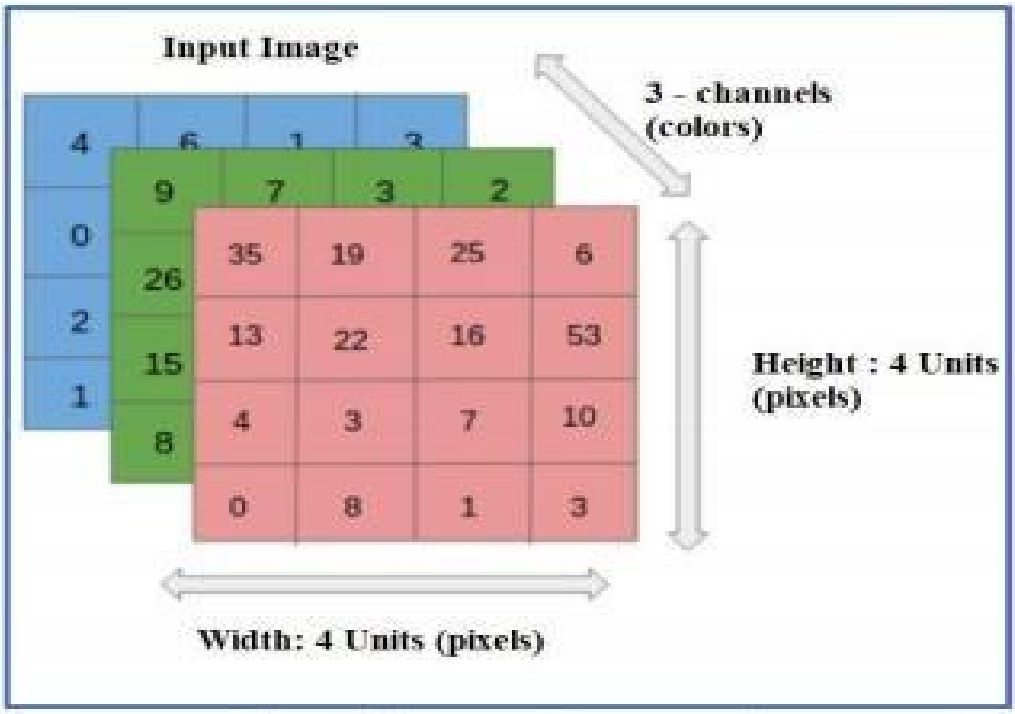
The entire procedure is separated into three phases, the first of which is data generation, analysis, and prediction, as shown in Figure 2.

Data from the patient, such as x-rays, mRI, and so on, has been generated utilising IoT devices. The information gathered may have come from photographs or from sensors for numerical data gathering, and it has been recorded in a database. These records can be viewed by authorised individuals at any time and from any location. Medical specialists analyse the data and use various algorithms to forecast cancer classifications.

We are utilising the CNN algorithm for analysis and prediction in this proposed model. It is vital to understand the many levels and adjustments made in each layer in order to comprehend the rest of the process. For example, as seen in Figure 2 input data should be in pixel format, which will be converted to a 3X3 image matrix in the conv 2D layer, and then to a single linear vector for classification.

Subject to appropriate channels, the CNN can obtain spatio-transient information. On the images, a planning communication is used to better comprehend the full visual information. The 4 x 4 x 3 picture is showed up in Figure (matrix representation)

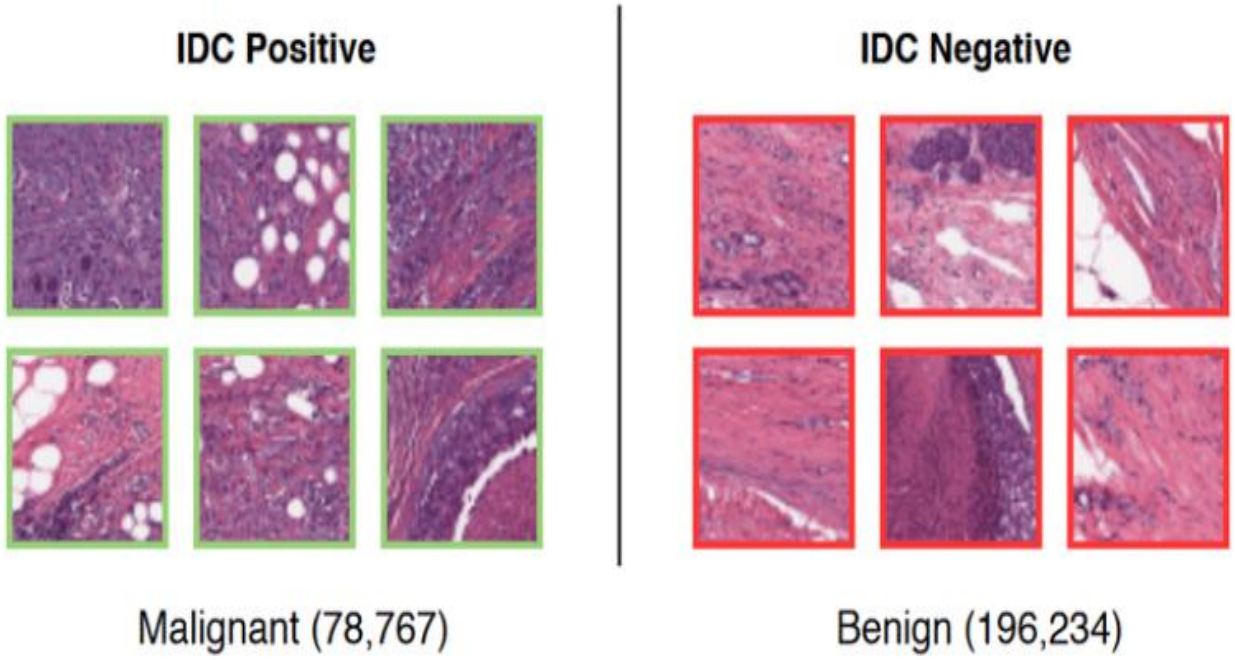




Pooled images are flattened into long vector which is fed to fully connected ANN. An artificial neural association's core design consists of a large number of interconnected neurons grouped in three layers: input, concealment, and yield. By considering enough models, this type of association usually sorts out a way to do tasks. The following is a representation of forward inducing and a single neuron:

$$\text{Rectified Linear Unit: Activation}(x) = \begin{cases} 0, \text{ for } x \leq 0 \\ x, \text{ for } x > 0 \end{cases}$$

$$\text{Softmax: Activation}(x) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}, \text{ where } i = 1, 2, \ldots, j$$



Through randomized rotation, reflection, and translation, our augmented dataset increased by a factor of 3, increasing the total number of 50x50 images patches to be 825,003 (588,702 IDC negative and 236,301 IDC positive

Mammogram: An x-ray photo of the breast

- Using CNN we trained it to recognize what a healthy mammogram looks like then what a and mammogram look like.
- A healthy mammogram look like by seeing what the features or features family come and not healthy mammogram.

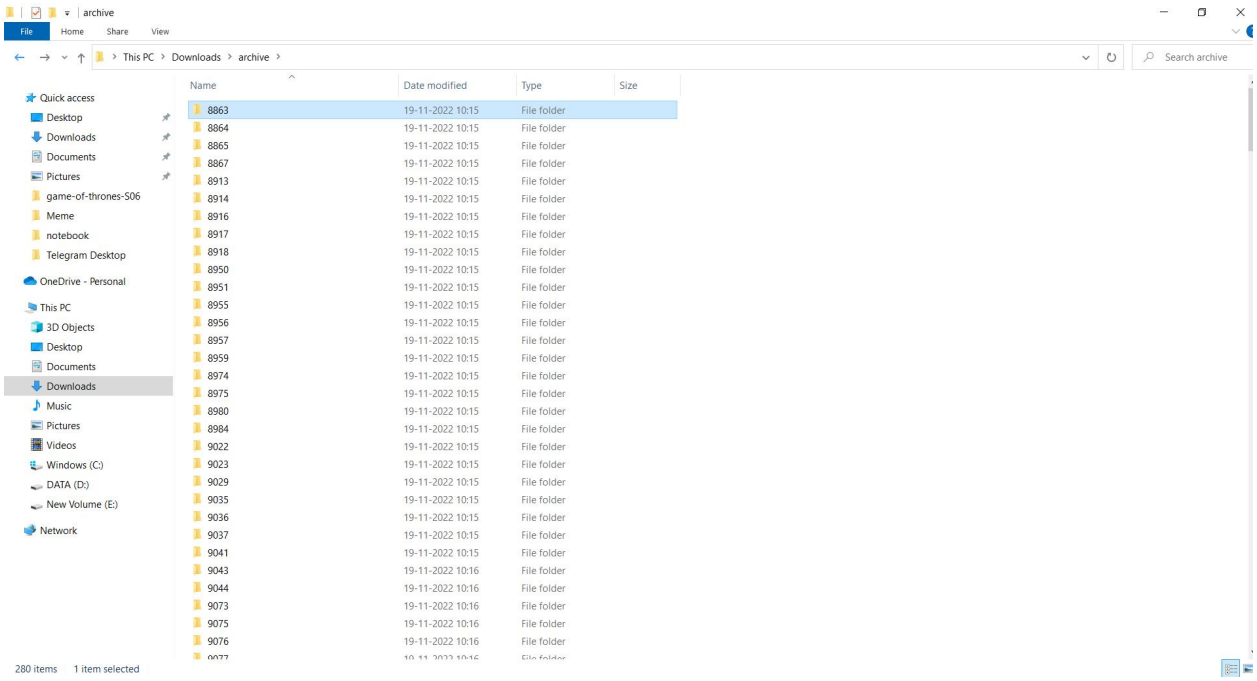
- For example -white blobs are white lumps in a breast would indicate a tumor or possibility of breast cancer for a pathologist

Models Imported

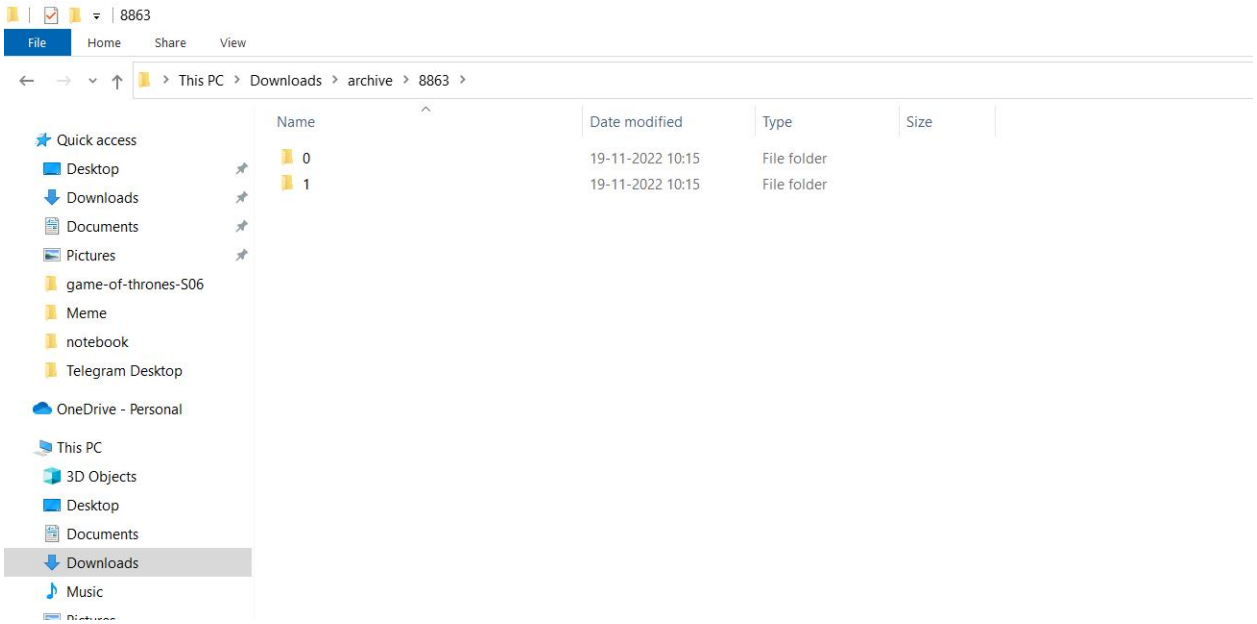
The modules that are used in this project are:

- Pandas
- Numpy
- Tensorflow
- os
- Imageio
- Itertools
- Shutil
- Matplotlib
- Seaborn
- cv2

Folder Structure



In this there are multiple folders, and in each folder there are two another folders labeled as 1 and 0. In which 1 has the cancer dataset and 0 has non-cancerous dataset.



CODE

```
importing necessary libraries

from numpy.random import seed
seed(101)

import pandas as pd
import numpy as np

import tensorflow

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout,
Conv2D, MaxPooling2D, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import
categorical_crossentropy
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping,
ReduceLROnPlateau, ModelCheckpoint

import os
from os import listdir
import cv2

import imageio
import skimage
import skimage.io
import skimage.transform

from sklearn.utils import shuffle
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from skimage.io import imread
import itertools
import shutil
import matplotlib.pyplot as plt
```

```
import seaborn as sns
%matplotlib inlin
```

```
# Removing duplicate folders to save space before running
the notebook

shutil.rmtree('/kaggle/working/all_images_dir',
ignore_errors=True)
shutil.rmtree('/kaggle/working/base_dir',
ignore_errors=True)
```

```
# Exploring the data structure

files = listdir("../input/breast-histopathology-images/")
print(len(files))
```

```
#looking at first 10 folders
files[0:10]
```

output:

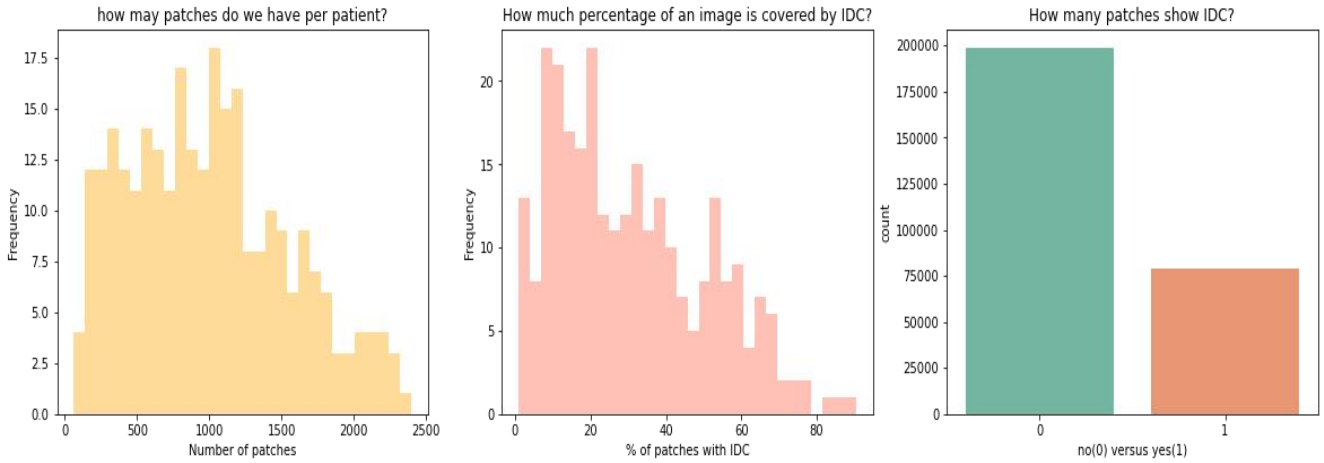
```
['13689',
 '12872',
 '8957',
 '14321',
 '12933',
 '8950',
 '9320',
 '8974',
 '14213',
 '14189']
```

```
#  We have to find the number of total images in the
dataset

total_images = 0
for n in range(len(folder)):
    patient_id = folder[n]
    for c in [0, 1]:
        patient_path = base_path + patient_id
        class_path = patient_path + '/' + str(c) + '/'
        subfiles = listdir(class_path)
        total_images += len(subfiles)

print("Total Images in dataset: ", total_images )
```

```
data = pd.DataFrame(index=np.arange(0, total_images),
columns=["patient_id", "path", "target"])

k = 0
for n in range(len(folder)):
    patient_id = folder[n]
    patient_path = base_path + patient_id
    for c in [0,1]:
        class_path = patient_path + "/" + str(c) + "/"
        subfiles = listdir(class_path)
        for m in range(len(subfiles)):
            image_path = subfiles[m]
            data.iloc[k]["path"] = class_path +
image_path
            data.iloc[k]["target"] = c
            data.iloc[k]["patient_id"] = patient_id
            k += 1

data.head()
```

```
cancer_perc =
data.groupby("patient_id").target.value_counts() /
data.groupby("patient_id").target.size()
canxer_perc = cancer_perc.unstack()
```

```
fig, ax = plt.subplots(1, 3,figsize = (20,5))
sns.distplot(data.groupby('patient_id').size(), ax=ax[0],
color='Orange', kde=False, bins=30)
ax[0].set_xlabel('Number of patches')
ax[0].set_ylabel('Frequency')
ax[0].set_title('how may patches do we have per patient?')
sns.distplot(cancer_perc.loc[:, 1]*100, ax=ax[1],
color="Tomato", kde=False, bins=30)
ax[1].set_title("How much percentage of an image is
covered by IDC?")
ax[1].set_ylabel("Frequency")
ax[1].set_xlabel("% of patches with IDC");
sns.countplot(data.target, palette="Set2", ax=ax[2]);
ax[2].set_xlabel("no(0) versus yes(1)")
ax[2].set_title("How many patches show IDC?");
```



Insights

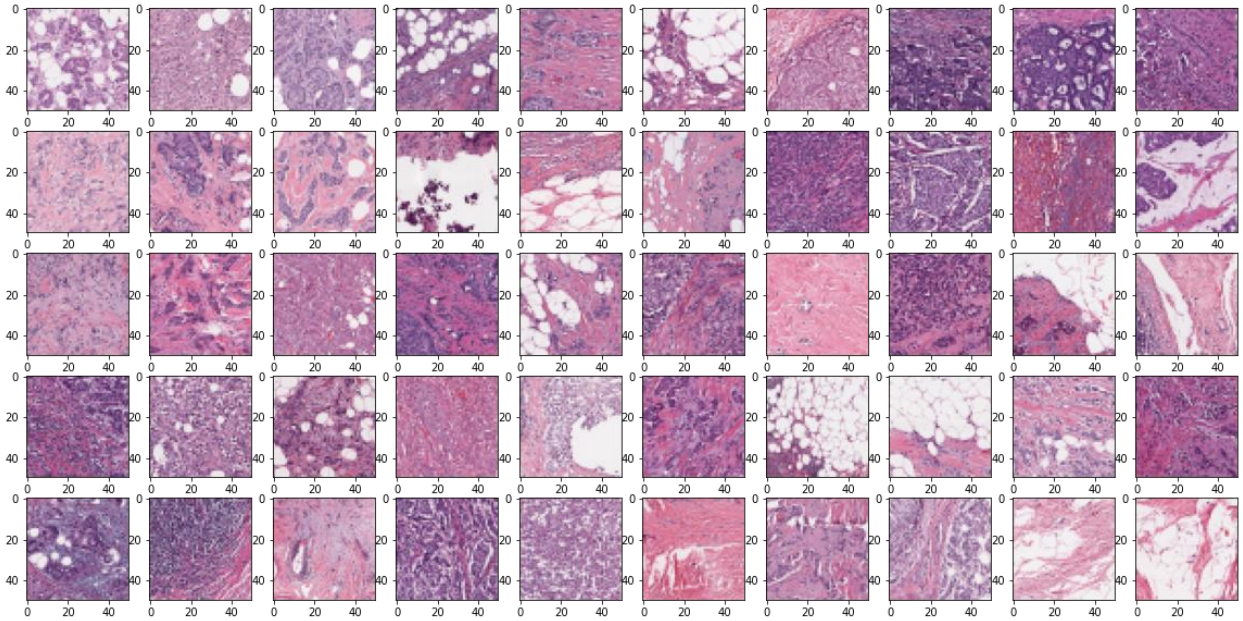
* The number of image patches per patient varies a lot.

* Some patients have more than 80 % patches that show IDC! Consequently the tissue is full of cancer or only a part of the breast was covered by the tissue slice that is focused on the IDC cancer.

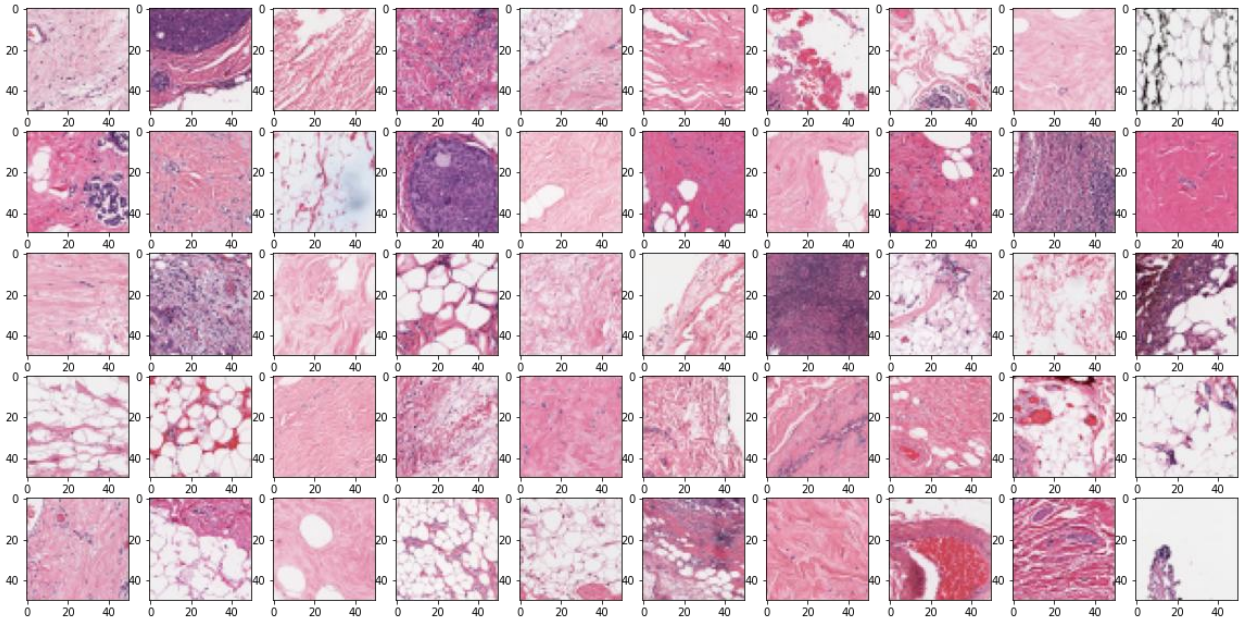
* The classes of IDC versus no IDC are imbalanced.

```
# coverting target to int

data.target = data.target.astype(np.int)
```

```
# displaying cancer set

cancer_selection = np.random.choice(data[data.target ==
1].index.values, size=50, replace=False)

fig, ax = plt.subplots(5, 10, figsize=(20, 10))

for n in range(5):
    for m in range(10):
        idx = cancer_selection[m + 10*n]
        image = imread(data.loc[idx, "path"])
        ax[n,m].imshow(image)
        ax[n,m].grid(False)
```



```
# displaying non-cancer set

non_cancer_selection = np.random.choice(data[data.target
== 0].index.values, size=50, replace=False)

fig, ax = plt.subplots(5, 10, figsize=(20, 10))

for n in range(5):
    for m in range(10):
        idx = non_cancer_selection[m + 10*n]
        image = imread(data.loc[idx, "path"])
        ax[n,m].imshow(image)
        ax[n,m].grid(False)
```



Insights

* Cancer Tissues appears to be more violet.

* But some non-caner tissue is also violet.

```
# Creating diresctory to store all images
all_images_dir = 'all_images_dir'

if os.path.isdir(all_images_dir):
    pass
else:
    os.mkdir(all_images_dir)
```

```
# This code copies all images from their seperate folders
into the same
# folder called all_images_dir.

'''
The directory structure is like:
    patient_id:
                0
                1
```

```
'''

patient_list = folder

for patient in patient_list:

    path_0 = "../input/breast-histopathology-
images/IDC_regular_ps50_idx5/" + str(patient) + '/0'
    path_1 = "../input/breast-histopathology-
images/IDC_regular_ps50_idx5/" + str(patient) + '/1'

    # create list of all files in folder 0
    file_list_0 = listdir(path_0)

    #create a list of all files in folder 1
    file_list_1 = listdir(path_1)

    # moving the 0 class images to all_images_dir
    for fname in file_list_0:

        src = os.path.join(path_0, fname)
        dst = os.path.join(all_images_dir, fname)
        shutil.copyfile(src, dst)

    # moving the 1 class images to all_images_dir
    for fname in file_list_1:

        src = os.path.join(path_1, fname)
        dst = os.path.join(all_images_dir, fname)
        shutil.copyfile(src, dst)
```

```
# creating dataframes for all images

image_list = os.listdir('all_images_dir')
df_data = pd.DataFrame(image_list, columns=['image_id'])

df_data.head()
```

	Image_id
1	13403_idx5_x2101_y01_class1.png
2	904_idx5_x2101_y90_class1.png
3	5465_idx5_x2101_y9701_class1.png
4	1554_idx5_x2101_y971_class0.png
5	995_idx5_x2101_y41_class0.png

```
# Defining helper functions

def extract_patient_id(x):

    a = x.split('_')
    patient_id = a[0]

    return patient_id

def extract_target(x):

    a = x.split('_')
    b = a[4]
    target = b[5]

    return target

# creating new column named patient_id
df_data['patient_id'] =
df_data['image_id'].apply(extract_patient_id)

#creating new column named target
df_data['target'] =
df_data['image_id'].apply(extract_target)
```

```
df_data.head(10)
```

	Image_id	Patient_id	Target
0	13403_idx5_x2101_y901_class1.png	13403	1
1	9324_idx5_x1001_y201_class1.png	9324	1
2	9325_idx5_x1751_y451_class0.png	9325	0
3	12954_idx5_x2701_y401_class0.png	12954	0
4	12905_idx5_x1151_y2001_class0.png	12905	0
5	12910_idx5_x951_y701_class1.png	12910	1
6	10272_idx5_x1951_y2101_class0.png	10272	0
7	13689_idx5_x1501_y1951_class0.png	13689	0
8	14211_idx5_x1201_y451_class1.png	14211	1
9	15840_idx5_x2351_y551_class0.png	15840	0

```
# class distribution of the images

df_data['target'].value_counts()
```

```
0    198738
1     78786
Name: target, dtype: int64
```

Balance the class distribution

* We can see that the class 1 images are higher in number that of class

* So to prevent this we balance the dataset

* We do this so that the Neural Network does not lean on favouring only one class

```
SAMPLE_SIZE = 78786

# take a sample of the majority class 0 (total = 198738)
df_0 = df_data[df_data['target'] ==
'0'].sample(SAMPLE_SIZE, random_state=101)
# take a sample of class 1 (total = 78786)
df_1 = df_data[df_data['target'] ==
'1'].sample(SAMPLE_SIZE, random_state=101)

# concat the two dataframes
df_data = pd.concat([df_0, df_1],
axis=0).reset_index(drop=True)

# Check the new class distribution
df_data['target'].value_counts()
```

```
y = df_data['target']

df_train, df_val = train_test_split(df_data,
test_size=0.10, random_state=101, stratify=y)

print(df_train.shape)
print(df_val.shape)
```

```
# Creating new base directory
```

```
base_dir ='base_dir'
os.mkdir(base_dir)

# Creating train directory inside base directory
train_dir = os.path.join(base_dir, 'train_dir')
os.mkdir(train_dir)

# Creating validation directory inside base directory
val_dir = os.path.join(base_dir, 'val_dir')
os.mkdir(val_dir)

# create new folders inside train_dir
a_no_idc = os.path.join(train_dir, 'a_no_idc')
os.mkdir(a_no_idc)
b_has_idc = os.path.join(train_dir, 'b_has_idc')
os.mkdir(b_has_idc)


# create new folders inside val_dir
a_no_idc = os.path.join(val_dir, 'a_no_idc')
os.mkdir(a_no_idc)
b_has_idc = os.path.join(val_dir, 'b_has_idc')
os.mkdir(b_has_idc)
```

```
# check that the folders have been created
os.listdir('base_dir/train_dir')
```

```
# Set the id as the index in df_data
df_data.set_index('image_id', inplace=True)
```

```
train_list = list(df_train['image_id'])
val_list = list(df_val['image_id'])

# Transfering the train images
for image in train_list:

    try:
        fname = image
```

```
        target = df_data.loc[image, 'target']

        if target == '0':
            label = 'a_no_idc'
        if target == '1':
            label = 'b_has_idc'

        # source path to image
        src = os.path.join(all_images_dir, fname)
        # destination path to image
        dst = os.path.join(train_dir, label, fname)
        # move the image from the source to the
destination
        shutil.move(src, dst)
    except:
        continue

for image in val_list:

    try:
        fname = image
        target = df_data.loc[image,'target']

        if target == '0':
            label = 'a_no_idc'
        if target == '1':
            label = 'b_has_idc'


        # source path to image
        src = os.path.join(all_images_dir, fname)
        # destination path to image
        dst = os.path.join(val_dir, label, fname)
        # move the image from the source to the
destination
        shutil.move(src, dst)

    except:
        continue
```

```
# check how many val images we have in each folder

print(len(os.listdir('base_dir/train_dir/a_no_idc')))
print(len(os.listdir('base_dir/train_dir/b_has_idc')))
```

```
train_path = 'base_dir/train_dir'
valid_path = 'base_dir/val_dir'

num_train_samples = len(df_train)
num_val_samples = len(df_val)
train_batch_size = 10
val_batch_size = 10

train_steps = np.ceil(num_train_samples /
train_batch_size)
val_steps = np.ceil(num_val_samples / val_batch_size)
```

```
datagen = ImageDataGenerator(rescale = 1.0 / 255,
                             rotation_range = 90,
                             zoom_range = 0.2,
                             horizontal_flip=True,
                             vertical_flip=True)

train_gen = datagen.flow_from_directory(train_path,
                                        target_size=(IMAG
E_SIZE,IMAGE_SIZE),
                                        batch_size=train_
batch_size,
                                        class_mode='categ
orical')

val_gen = datagen.flow_from_directory(valid_path,
                                        target_size=(IMAG
E_SIZE,IMAGE_SIZE),
                                        batch_size=val_ba
tch_size,
```

```
                                                class_mode='categ
orical')

# Note: shuffle=False causes the test dataset to not be
shuffled
test_gen = datagen.flow_from_directory(valid_path,
                                                target_size=(IMAG
E_SIZE,IMAGE_SIZE),
                                                batch_size=1,
                                                class_mode='categ
orical',
                                                shuffle=False)
```

```
# Building the model
kernel_size = (3,3)
pool_size= (2,2)
first_filters = 32
second_filters = 64
third_filters = 128

dropout_conv = 0.3
dropout_dense = 0.3


model = Sequential()
model.add(Conv2D(first_filters, kernel_size, activation =
'relu',
                 input_shape = (IMAGE_SIZE, IMAGE_SIZE,
3)))
model.add(Conv2D(first_filters, kernel_size, activation =
'relu'))
model.add(Conv2D(first_filters, kernel_size, activation =
'relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))

model.add(Conv2D(second_filters, kernel_size, activation
='relu'))
```

```
model.add(Conv2D(second_filters, kernel_size, activation
='relu'))
model.add(Conv2D(second_filters, kernel_size, activation
='relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))

model.add(Conv2D(third_filters, kernel_size, activation
='relu'))
model.add(Conv2D(third_filters, kernel_size, activation
='relu'))
model.add(Conv2D(third_filters, kernel_size, activation
='relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))

model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(dropout_dense))
model.add(Dense(2, activation = "softmax"))

model.summary()
```

```
# Training the model
model.compile(Adam(lr=0.0001), loss='binary_crossentropy',
              metrics=['accuracy'])
```

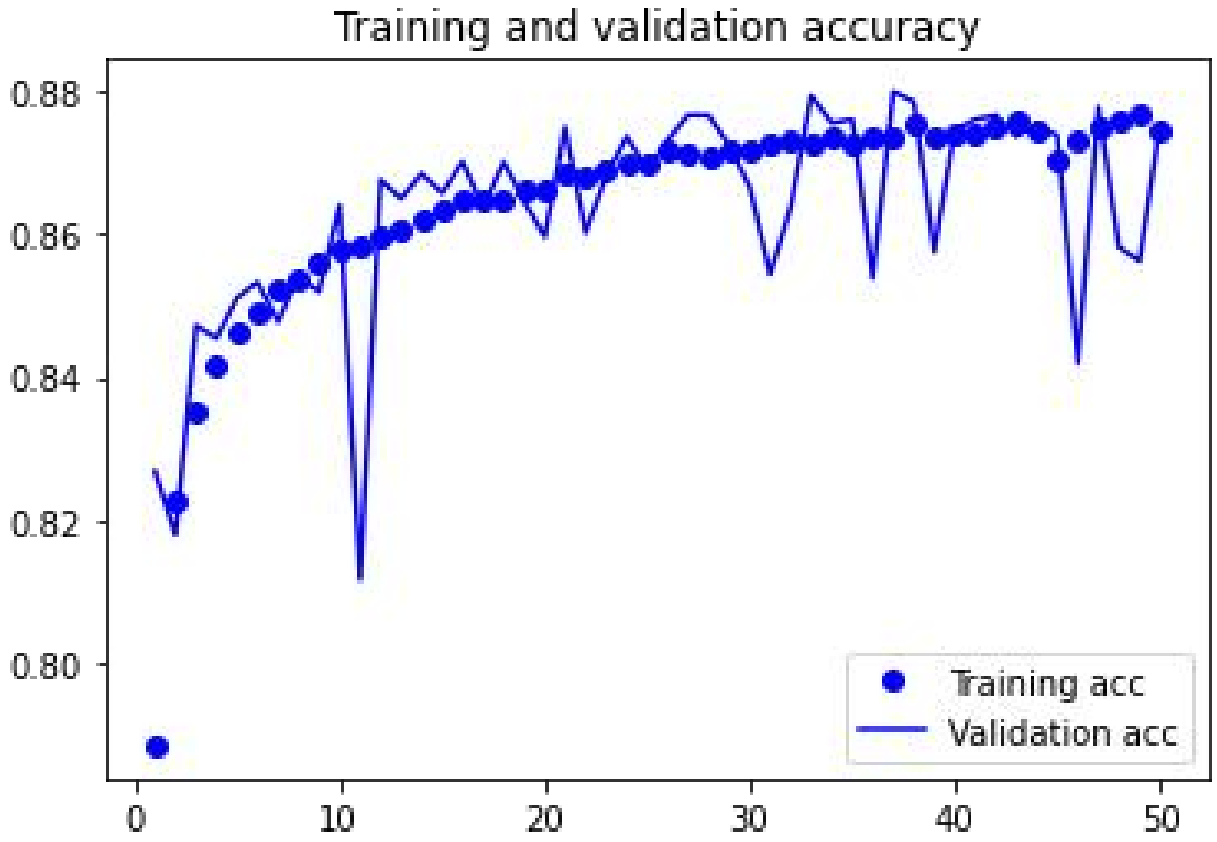
```
filepath = "model.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_acc',
verbose=1,
                             save_best_only=True,
mode='max')

reduce_lr = ReduceLROnPlateau(monitor='val_acc',
factor=0.5, patience=3,
                                    verbose=1, mode='max',
min_lr=0.00001)
```

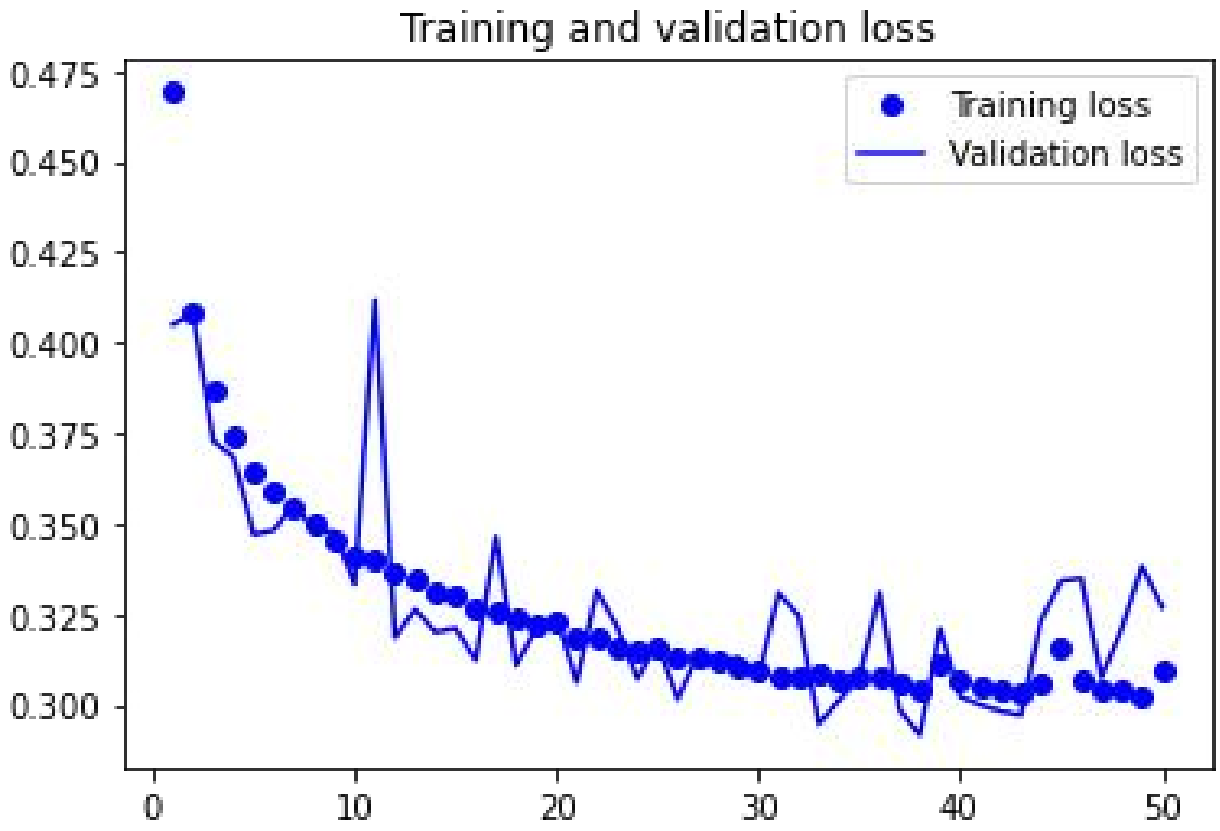
```
callbacks_list = [checkpoint, reduce_lr]

history = model.fit_generator(train_gen,
steps_per_epoch=train_steps,
                    validation_data=val_gen,
                    validation_steps=val_steps,
                    epochs=50, verbose=1,
                   callbacks=callbacks_list)

try:
    model.save('/kaggle/working/model.h5')
except:
    pass

try:
    model.save('model.h5')
except:
    pass
```

```
racy: 0.8752
Epoch 45/50
14182/14182 [==============================] - 321s 23ms/step - loss: 0.3155 - accuracy: 0.8704 - val_loss: 0.3342 - val_accu
racy: 0.8737
Epoch 46/50
14182/14182 [==============================] - 321s 23ms/step - loss: 0.3072 - accuracy: 0.8730 - val_loss: 0.3349 - val_accu
racy: 0.8420
Epoch 47/50
14182/14182 [==============================] - 325s 23ms/step - loss: 0.3041 - accuracy: 0.8748 - val_loss: 0.3076 - val_accu
racy: 0.8777
Epoch 48/50
14182/14182 [==============================] - 323s 23ms/step - loss: 0.3037 - accuracy: 0.8756 - val_loss: 0.3210 - val_accu
racy: 0.8582
Epoch 49/50
14182/14182 [==============================] - 324s 23ms/step - loss: 0.3025 - accuracy: 0.8767 - val_loss: 0.3383 - val_accu
racy: 0.8562
Epoch 50/50
14182/14182 [==============================] - 329s 23ms/step - loss: 0.3097 - accuracy: 0.8746 - val_loss: 0.3269 - val_accu
racy: 0.8749
```

```
# display the loss and accuracy curves

import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.figure()

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
```

Output:


Training and validation accuracy
0.88
0.86
0.84
0.82
0.80
0
10
20
30
40
50
Training acc
Validation acc




```
### Make a prediction on the val set
We need these predictions to calculate the AUC score,
print the Confusion Matrix and calculate the F1 score.

predictions = model.predict_generator(test_gen,
steps=len(df_val), verbose=1)
```

```
# Put the predictions into a dataframe.
# The columns need to be oredered to match the output of
the previous cell

df_preds = pd.DataFrame(predictions, columns=['no_idc',
'has_idc'])

df_preds.head()
```

	no_idc	has_idc
0	0.690448	0.309552
1	0.685284	0.314716
2	0.293493	0.706507
3	0.332423	0.667577
4	0.829533	0.170467

```
# Get the true labels
y_true = test_gen.classes

# Get the predicted labels as probabilities
y_pred = df_preds['has_idc']
```

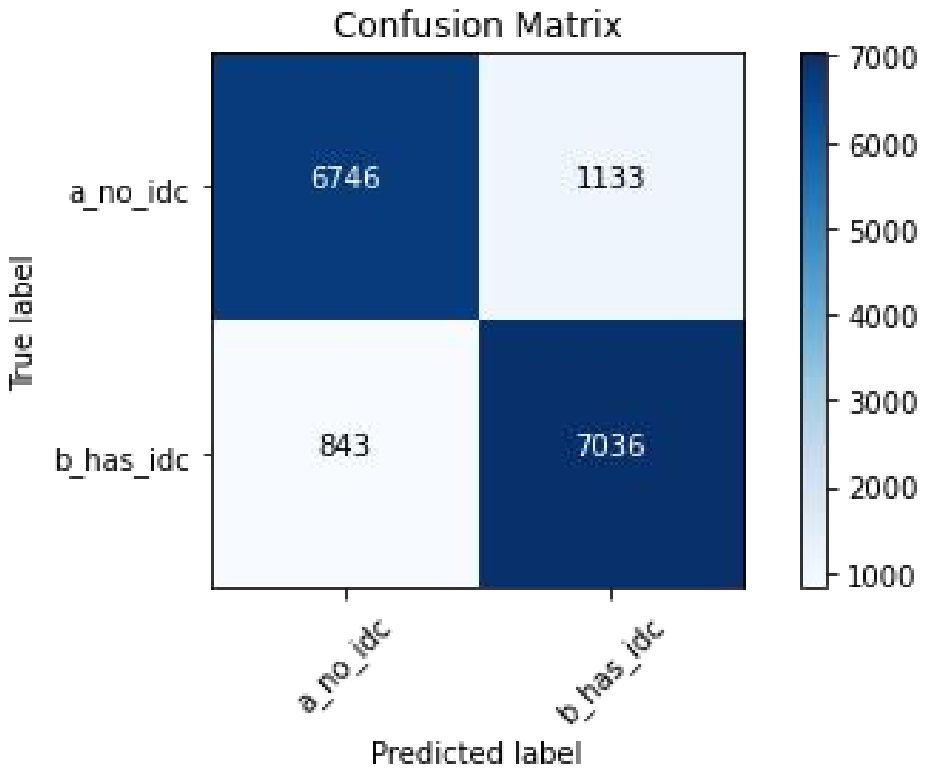
```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_true, y_pred)
```

Output:

```
0.9459758550448937
```

```
# Creating the confusion matrix
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion Matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:,
np.newaxis]
```

```
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else
"black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
```


Confusion Matrix
6746
1133
843
7036
a_no_idc
b_has_idc
True label
a_no_idc
b_has_idc
Predicted label
7000
6000
5000
4000
3000
2000
1000

Conclusion

The main aim of this model is to provide the earlier warning to the users and it is also cost and time saving benefit to the user. The mortality rate of breast cancer is the highest among all other types of cancer; it can be detected early by detecting the breast nodules. In this system, image pre-processing and image segmentation are implemented to obtain the diagnosis result. By using these steps, the nodules are detected and some features are extracted. The extracted features can be used for classification of disease stages. Determining the nodule features can provide to know more information of the condition of breast cancer at the early stages. This technique helps the radiologists and the doctors by providing more information and taking correct decision for breast cancer patient in short time with accuracy. Therefore, this method is less costly. less time consuming and 13:33/14:53 easy to implement.

Results and Discussion

Experiment Configuration The experiment is run on a 64-bit dual core machine with a processor speed of 1.99GHz using 1GB of DDR2 RAM. The research is run using Python since all of the algorithms are coded in Python. All images are converted into grayscale. We opt for grayscale since it is simpler to handle. We coded an algorithm to detect whether an image belong to IDC group or Non IDC group.

References

[1] Noone AM, Howlader N, Krapcho M, Miller D, Brest A, Yu M, Ruhl J, Tatalovich Z, Mariotto A, Lewis
DR, Chen HS, Feuer EJ, and Cronin KA. Seer cancer statistics review, 1975-2015. Technical report,
National Cancer Institute., Nov 2017.

[2] Choudhury, Avishek, Perumalla, Sunanda, Detecting breast cancer using artificial intelligence:
Convolutional neural network, *Technology and Health Care*, pp. 1-11, 2020.

[3] https://www.kaggle.com/paultimothymooney/breast-histopathology-images

[4] American Cancer Society. Breast Cancer Facts & Figures 2017-2018. Atlanta: American Cancer Society, Inc. 2017.

[5] H. Chapala and B. Sujatha, ResNet: Detection of Invasive Ductal Carcinoma in Breast Histopathology Images Using

Deep Learning, *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*