BREAST CANCER DETECTION USING CONVOLUTIONAL NEURAL NETWORK

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

SHRUSHTI GUPTA, 121810305020

SHASHANK ALA, 121810305010

ROHITH REDDY, 121810305049

RASAMSETTI SAMPATH, 121810305058

CHINTA SUMANTH, 121810305028

Under the esteemed guidance of

Dr. Radhika Y

Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM

(Deemed to be University)

VISAKHAPATNAM

March 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY GITAM

(Deemed to be University)

DECLARATION

We, hereby declare that the project report entitled "**BREAST CANCER DETECTION USING CONVOLUTIONAL NEURAL NETWORK**" is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 1st April, 2021.

Registration No(s).	Name(s)	Signature(s)
121810305020	**SHRUSHTI GUPTA**	
121810305010	**SHASHANK ALA**	
121810305049	**ROHITH REDDY**	
121810305058	**RASAMSETTI SAMPATH**	
121810305028	**CHINTA SUMANTH**	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM INSTITUTE OF TECHNOLOGY GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled "**BREAST CANCER DETECTION USING CONVOLUTIONAL NEURAL NETWORK**" is a bonafide record of work carried out by **SHRUSHTI GUPTA (121810305020), SHASHANK ALA (121810305010), ROHITH REDDY (121810305049), RASAMSETTI SAMPATH (121810305058), CHINTA SUMANTH (121810305028)** students submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

Dr. Radhika Y	Dr. Sireesha R
Professor	Professor

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from our guide and we are extremely privileged to have got this all along with the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank our guide

We respect and thank Dr. Radhika Y ma'am, for enlightening us for "**BREAST CANCER DETECTION USING CONVOLUTIONAL NEURAL NETWORK"** and giving us all support and guidance, which made us complete the project duly. We are extremely thankful to ma'am for providing such nice support and guidance even while having a busy schedule.

TABLE OF CONTENTS

1. ABSTRACT

Image processing algorithms are commonly employed in various medical domains to increase disease detection capability. Cancer is one of the most dangerous diseases, and early detection is a challenging task in medicine. The project is based on breast cancer samples. After a comparative examination of commonly used methods in each category, an appropriate and efficient approach is adopted in each of the design processes of the proposed framework. Typically, the cancer detection method entails classifying the image biopsy as malignant or benign. Many abnormalities are observed and classified in microscopic biopsy image analysis by doctors and pathologists based on various characteristics of the cell nuclei, such as shape, size, and proportion of nucleus to the cytoplasm.

2. INTRODUCTION

Cancer detection has always been a severe challenge in diagnosis and treatment planning for pathologists and healthcare professionals. Manual cancer detection using microscopic biopsy images is subjective; results may differ from expert to expert based on their experience and other factors such as the lack of specific and reliable quantitative metrics to classify biopsy images as normal or cancerous. When biologically interpretable and clinically meaningful feature-based techniques are applied for disease detection, the automated identification of malignant tissue from microscopic biopsy pictures helps to alleviate the difficulties mentioned above and provides improved outcomes. Many models have been developed and deployed in recent years, each with benefits and drawbacks. Histopathologists examine the particular traits in the cells and tissue architecture to discover and diagnose cancer from microscopic biopsy photos. Shape and size of cells, shape and size of cell nuclei, and cell distribution are frequent parameters used for cancer detection and diagnosis from microscopic biopsy images [1].

3. LITERATURE REVIEW

Several recent studies have applied image processing techniques and different classification algorithms on numerous medical datasets to classify tissue (benign vs. malignant).

"**Boosting Breast Cancer Detection Using Convolutional Neural Network**," written by "Saad Awadh Alanazi, M. M. Kamruzzaman, Md Nazirul Islam Sarker , Madallah Alruwaili Yousef Alhwaiti , Nasser Alshammari , and Muhammad Hameed Siddiqi" in 2021, is the chosen base paper.

[1] This paper investigates the proposed system, which uses multiple convolutional neural network (CNN) architectures to automatically identify breast cancer, compared to those of machine learning (ML) techniques. An extensive collection of roughly 275,000 50 50-pixel RGB picture patches guided all structures. For quantitative data, validation tests were conducted using the performance measures for each approach. The proposed CNN design is proven successful, with an accuracy rate of 87%. Using a secondary database like Kaggle is the study's biggest weakness, and future studies should be based on primary data for more accurate breast cancer detection outcomes.

[2] The proposed methodology includes several stages, including microscopic image enhancement, background cell segmentation, feature extraction, and finally, classification. The contrast limited adaptive histogram equalization method is used to highlight the features of the tissue and structures. The k-means segmentation algorithm is used to segment background cells because it performs better than other commonly used segmentation algorithms. Various physiologically interpretable and clinically relevant shapes and morphology-based features are proposed to be extracted from the segmented pictures during the feature extraction phase. Finally, the k-nearest neighborhood approach is employed for picture categorization into benign and malignant categories because it outperforms other commonly used methods in this application. The maximum accuracy, sensitivity, and specificity values are 0.9552, 0.9615, and 0.9543, respectively.

[3] This paper presents a learning-based system that segregates glands in benign and malignant colorectal cancer tissue. Two deep convolutional neural networks (CNN) are trained as pixel classifiers after preprocessing images according to the HematoxylinEosin staining technique. The CNN predictions are subsequently regularised using a figure-ground segmentation based on the weighted total variation to achieve the final segmentation result. The approach achieves a tissue

classification accuracy of 98% and 94% on two test sets. However, because there was no further information (e.g., high- or low-grade) provided in addition to the segmentation ground truth, it was unable to identify more detailed histologic grades among the instances.

[4] An accurate system can be proposed by implementing the critical phases of image processing to predict and detect cancerous lung tumors. As a result, Deep-Learning with Instantaneously Trained Neural Networks "DTINN" achieved the highest accuracy in classifying the lung-cancer nodule, with 98.42 %, using the Weighted Mean Histogram Cultural sensitivity "WMHE" approach as an image processing mechanism, followed by multi "SVM" classifier with 97 % using Grey level Co-Occurrence Matrix "GLCM." Future study plans to provide an efficient Lung-Cancer detection and prediction system by using image preprocessing procedures such as weighted median filter, watershed approach image segmentation, and feature extraction using swarm intelligence algorithms.

[5] A strategy for classifying benign and malignant cells in cytological pictures of the lungs has been proposed. Cropped microscopic images with a resolution of 224x224 pixels are utilized in this dataset (306 benign and 315 malignant images). A DCNN (Deep Convolutional Neural Network) classification system that is automated is proposed. The evaluation results' sensitivity, specificity, and accuracy were 89.3, 83.3, and 87.0 percent, respectively. The findings and analysis demonstrated that the proposed approach could detect cancerous cells in microscopic cytological images. The number of cases used to train DCNN will increase in future work to improve classification accuracy. Furthermore, it is possible to use DCNN with several layers.

4. PROBLEM IDENTIFICATION AND OBJECTIVES

- To classify Cancerous and Non-cancerous tissues using Image Processing and Machine Learning Techniques.
- To find out the accuracy of the classification with image processing and data augmentation.
- To compare the accuracy of various Machine Learning algorithms to check which one is the most accurate.

5. SYSTEM METHODOLOGY

5.1 The Dataset

The dataset used in this project is microscopic biopsy images of breast cancer[Fig.1] and microscopic biopsy images of non-cancerous cells[Fig.2] derived from the Breast Cancer Histopathological Database (BreakHis). The Breast Cancer Histopathological Image Classification (BreakHis) is made up of 1693 microscopic images of breast tumor tissue collected from 82 patients, 547 of which are benign and 1147 of which are malignant (700X460 pixels, 3-channel RGB, 8-bit depth in each channel, PNG format).

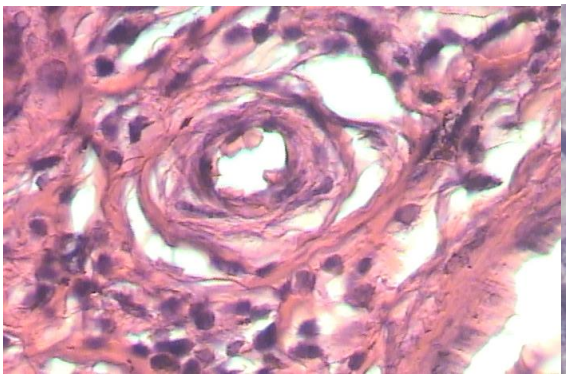

Fig.1: Cancerous tissue

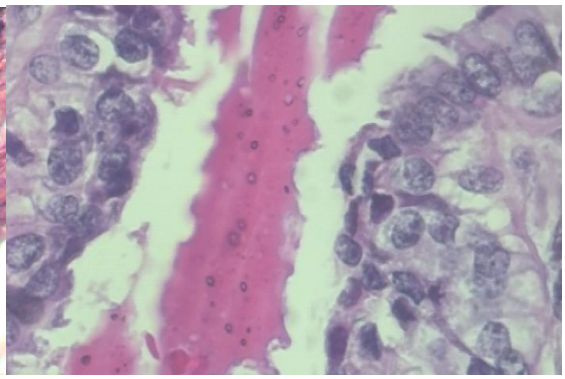

Fig.2: Non-Cancerous tissue

This database was created in partnership with Parana, Brazil's P&D Laboratory – Pathological Anatomy and Cytopathology. The SOB method, also known as partial mastectomy or excisional biopsy, was used to acquire the samples in the dataset. This operation extracts a larger tissue sample and is performed under general anesthesia in a hospital compared to other needle biopsy types.

5.2 Tools Used

Editor/Programming Tool: Google Colaboratory

Programming language used: Python3

5.3. Flowchart

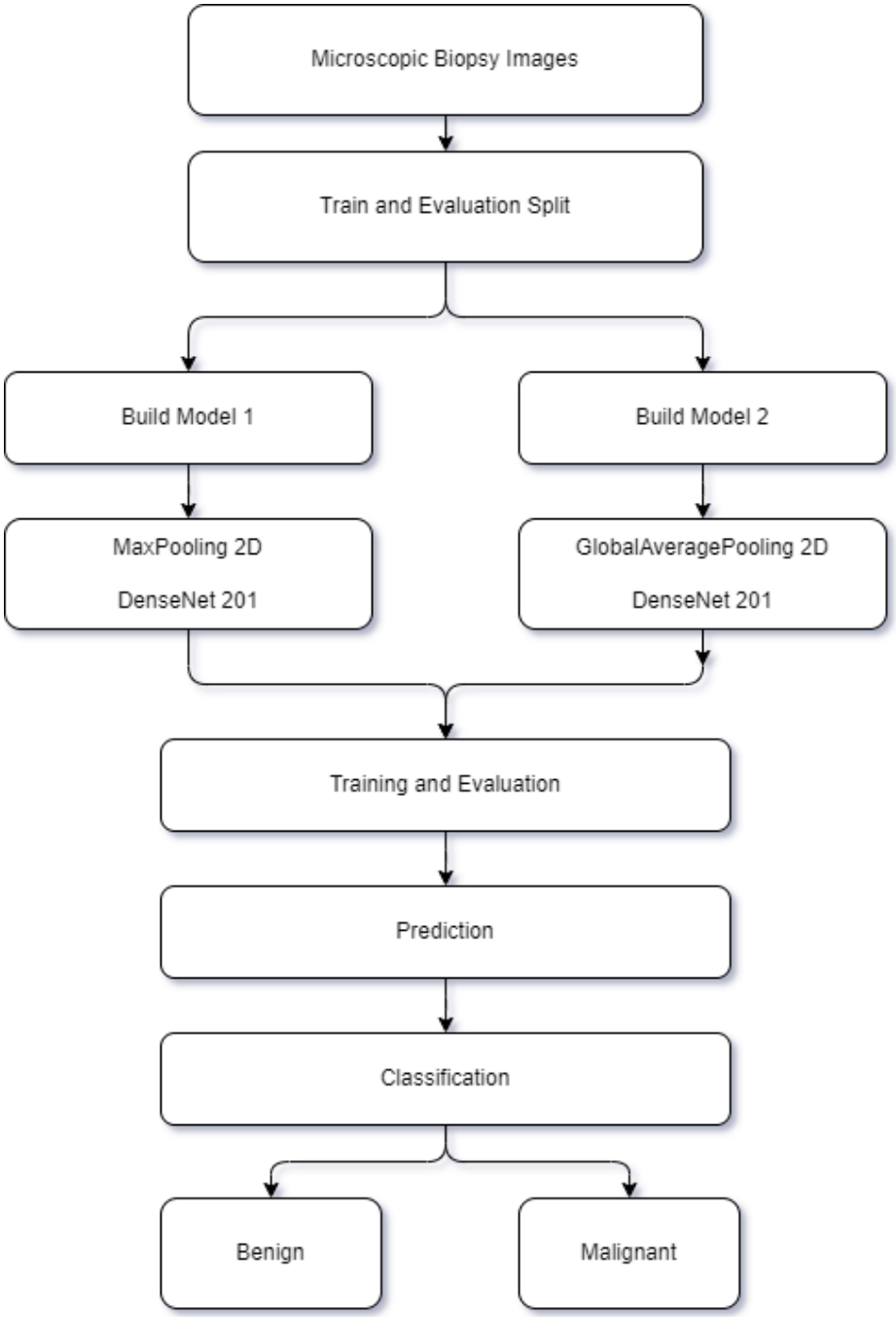

Fig.3: Flow of the project

6. OVERVIEW OF TECHNOLOGIES

6.1 Machine Learning

Machine learning (ML) is a branch of artificial intelligence (AI) that enables software products to improve their forecasting accuracy without being explicitly designed to do so. In order to forecast new output values, machine learning algorithms employ historical and statistical data as input.

Fig.4: Machine Learning flow
Source: Great Learning

To put it another way, machine learning is the process of computers discovering useful information without being directed where to seek. Instead, they use algorithms that learn from data in an iterative approach to accomplish this. As one feeds more data into a machine, the algorithms learn more about it, which improves the quality of the output. At its most basic level, machine learning is the ability to adapt to new data independently and iteratively. To create reliable and informed results, applications learn from prior computations and transactions and employ "pattern recognition."

Inputting training data into the chosen algorithm is the first step in the Machine Learning process. Training data must be known or unknown to construct the final Machine Learning algorithm. The type of training data used in the method affects the algorithm.

New input data is fed into it to see if the machine learning algorithm is working correctly. After then, the prediction and the results are compared.

If the prediction and the results do not match, the algorithm is re-trained until the data scientist achieves the desired result. This allows the machine learning algorithm to learn and produce the best response on its own, gradually improving in accuracy over time.

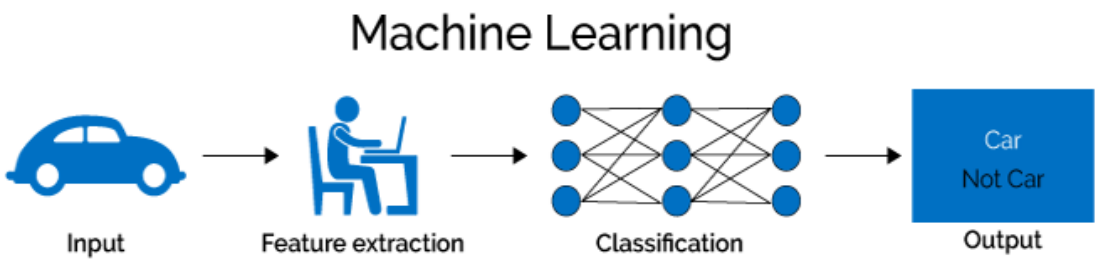

Fig.5: Machine Learning Example

Source: Internet

Python is the best programming language for machine learning because of the numerous libraries and widespread support. Python is suitable for data analysis and data mining since it supports a wide range of methods (such as classification, clustering, regression, dimensionality reduction) and machine learning models.

6.2 Image Processing

To comprehend image processing, one must first learn the concept of an image. A picture can be represented using the 2D function F(x,y), where x and y are spatial coordinates. The amplitude of F at a specific value of x,y equals the intensity of a picture at that place. The x, y, and amplitude values in a digital image are all finite. It's a pixel array divided into rows and columns. Pixels are image elements that store color and intensity information. The spatial coordinates x, y, and z can also be used to depict a picture in 3D Pixels are layered in a matrix format. An RGB image is what this is called.

There are various types of images

- RGB images
- Grayscale images

Image processing is the application of operations to an image to improve it or derive helpful information from it. It is a form of signal processing in which a picture serves as the input, and the output is either that image or its characteristics/features. Image processing is one of today's most rapidly changing technologies. It's also an important area of research in engineering and computer science.

The three phases that makeup image processing are as follows:

- Importing an image using image acquisition tools;
- Analyzing and altering the image;
- Producing an output that can be an altered image or a report based on image analysis.

Image Processing Sample:

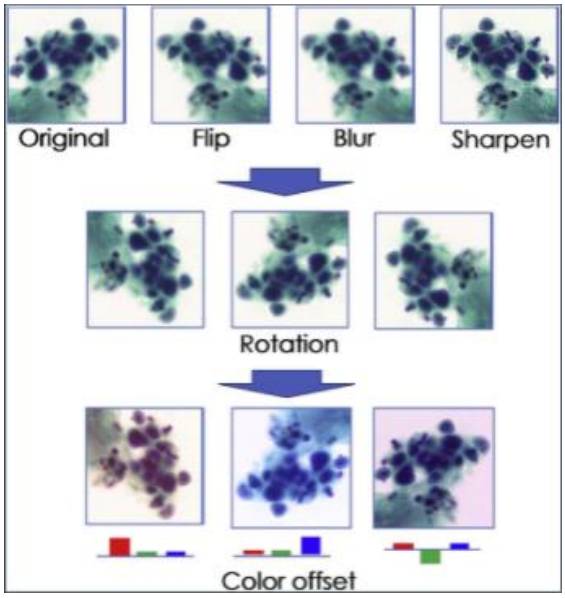

Fig.6: Different image processing samples Source [5]

6.3 Convolutional Neural Network

A CNN or Convolutional Neural Network is a deep learning neural network designed to analyze structured arrays of data-like representations. When one thinks about neural networks, one usually thinks of matrix multiplications, but this isn't the case with ConvNet. It employs a technique known as Convolution. Convolution is a mathematical operation on two functions that yields a third function that explains how the shape of one is changed by the other.

CNN's are excellent at detecting unique features in input images, such as lines, gradient circles, and even eyes and faces. Because of this feature, convolutional neural networks are effective in computer vision. CNN does not require pre-processing and can run straight on an under-done image. Feedforward neural network with up to 20 layers is known as a Convolutional Neural

Network. Convolutional Neural Network strength stems from a layer known as the convolutional layer. CNN is made up of multiple convolutional layers stacked on top of each other, capable of identifying more complex structures.

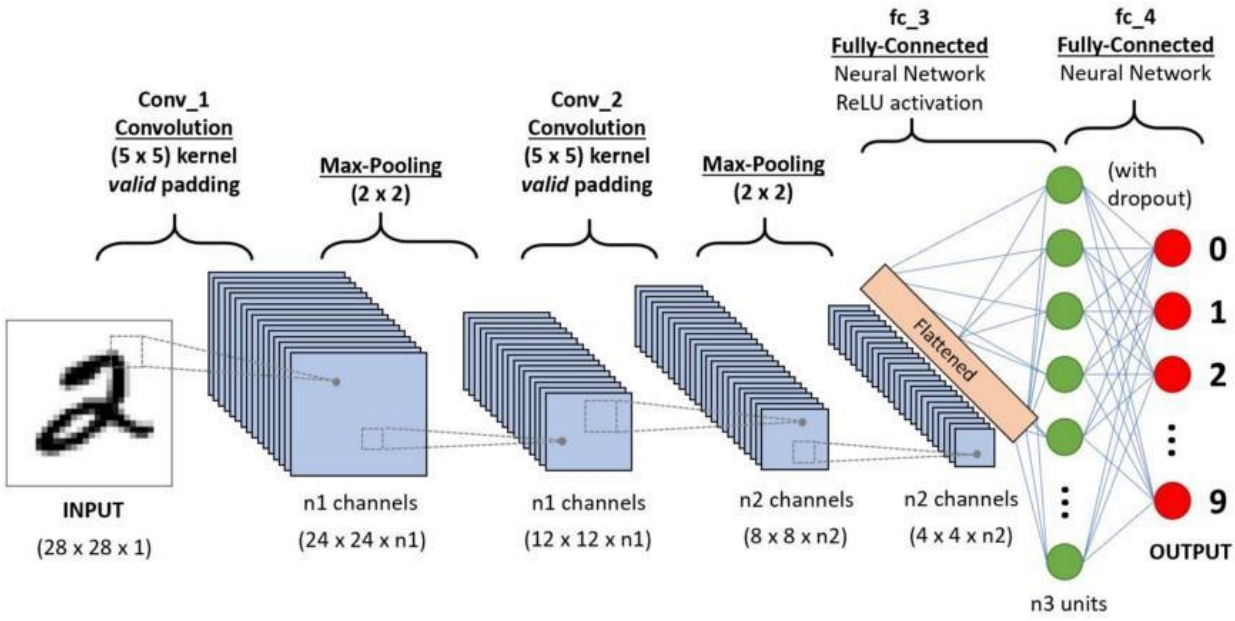

Fig.7: Convolutional Neural Network

Source: Analytics Vidhya

Convolutional layers, pooling layers, and fully-connected (FC) layers are the three types of layers that make up the CNN. A CNN architecture will be constructed when these layers are stacked.

The following are the three types of pooling operations:

- Max pooling: The batch's maximum pixel value is considered
- Min pooling: The batch's minimum pixel value is considered.
- Average pooling: It selects the average value of all pixels in the batch.

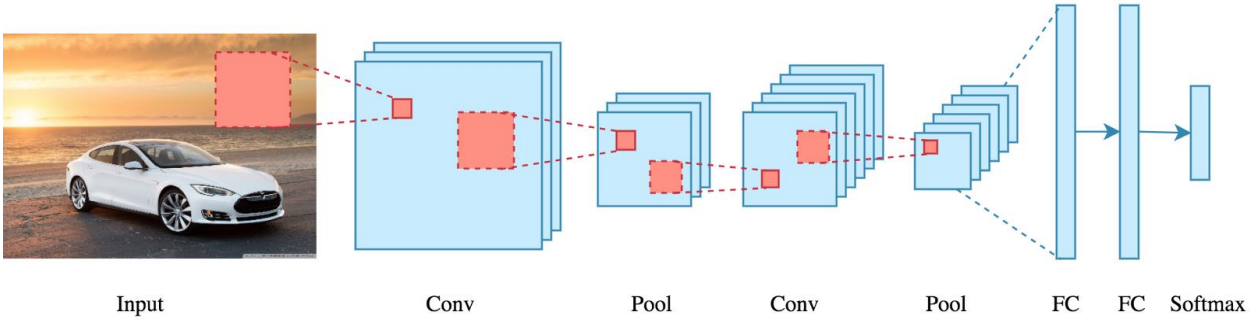

Fig.8: Convolutional Neural Network showing different layers

Source: Analytics Vidhya

DenseNet-201 is a 201-layer Convolutional Neural Network (CNN) that uses dense conditions between layers via Dense Blocks, which allow all layers (with matching feature-map sizes) to be directly connected. A pre-trained version of the network, trained on millions of photos, can classify images into 1000 object categories, including keyboards, pencils, and many human faces and animals.

In the end, ConvNet's job is to compress the images into a format that is easier to handle while preserving elements that are important for making a decent prediction. CNN delivers in-depth findings despite its power and resource complexity. It's all about spotting patterns and characteristics so small and insignificant that they go unnoticed by the human eye. However, when it comes to comprehending the contents of an image, it falls short.

6.4 Python Libraries and Modules

Numpy: NumPy is a Python library that allows one to work with arrays. NumPy functions can be used to perform image processing after reading the image as a NumPy array. Pixel values can be obtained and set (changed), images can be trimmed or concatenated, and so on. NumPy arrays, unlike lists, are stored in a single continuous location in memory, allowing programs to access and alter them quickly. In computer science, this is referred to as the locality of reference. This is the primary reason why NumPy outperforms lists. It's also been tweaked to work with the most recent

CPU architectures.

It is the most crucial Python package for scientific computing. It has a number of features, including the following:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Matplotlib: Matplotlib is a Python plotting library, and it is NumPy numerical math extension. It offers an object-oriented API for incorporating plots into programs written in general-purpose GUI toolkits like wxPython, Tkinter, Qt, or GTK. One of the most significant advantages of visualization is that it provides us with visual access to massive volumes of data in simply understandable graphics. Matplotlib has a variety of plots such as line, bar, scatter, histogram, and so on.

Scikit-image(skimage): Scikit-image (skimage) is a Python-based image processing package that is free and open-source. It includes image processing algorithms such as segmentation, filtering, modification, and feature recognition. Segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature recognition, are the different algorithms that are included in Scikit. It is compatible with the NumPy and SciPy Python numerical and scientific libraries.

Cv2: The OpenCV-Python library is a collection of Python bindings for dealing with computer vision problems. The method cv2.imread() opens a file and loads an image. If the image cannot be read, this method returns an empty matrix (due to insufficient permissions, a missing file, or an unsupported or invalid format).

Keras: Keras is a Python-based deep learning API that runs on top of TensorFlow. It was created

with the goal of allowing users to experiment quickly. Keras is based on a simple framework that makes it simple to build deep learning models using TensorFlow. Keras is a deep learning framework that allows one to define models quickly. It works with a variety of platforms and backends. It's an easy-to-use framework that works on both the CPU and the GPU.
Keras makes high-level neural network API easier and more performant by utilising multiple optimization approaches. It has the following capabilities:

- API that is consistent, straightforward, and extensible.
- Minimal structure - simple to get the desired outcome without any frills
- It works with a variety of platforms and backends.
- It's a simple framework that operates on both the CPU and the GPU.
- Computational scalability is high.

TensorFlow: TensorFlow is a machine learning and artificial intelligence software library that is free and open-source. It can be used for various applications, but it focuses on deep neural network training and inference. This adaptability lends itself to a wide range of applications in a variety of industries. Its adaptable architecture enables computing to be deployed over a wide range of platforms (CPUs, GPUs, TPUs), from PCs to server clusters to mobile and edge devices.

Scikit-learn: Scikit-learn (commonly known as sklearn) is a Python machine learning library available for free. It includes support-vector machines, random forests, gradient boosting, k-means, and DBSCAN, among other classification, regression, and clustering techniques. It is designed to work with the Python numerical and scientific libraries NumPy and SciPy.

Pandas: Pandas is a Python library that provides quick, versatile, and expressive data structures for working with "relational" or "labeled" data. Its goal is to serve as the foundation for undertaking realistic, real-world data analysis in Python. Furthermore, it aspires to be the most powerful and flexible open-source data analysis and manipulation tool available in any language.

Scipy: Optimization, integration, interpolation, eigenvalue issues, algebraic equations, differential equations, statistics, and many other types of problems are all covered by SciPy. It

expands NumPy by adding array computation capabilities as well as specific data structures like sparse matrices and k-dimensional trees. SciPy's high-level syntax makes it usable and productive for programmers of many backgrounds and levels of experience.

JSON: JSON stands for JavaScript Object Notation, a popular data interchange standard on the internet. JSON is the best format for storing and structuring data between clients and servers. It has a syntax that is comparable to that of JavaScript. JSON's primary goal is to transfer data between the client and the webserver.

Math: Python comes with a collection of built-in math functions and a comprehensive math module that allow users to execute mathematical operations on numbers.

OS: In Python, the OS module has functions for dealing with the operating system. Python's standard utility modules include OS. This module allows users to use operating system-dependent functions on the go. The os and os path modules include many functions to interface with the file system that are included in the os and os.path modules.

GC: This module offers access to the optional garbage collector (GC). It allows you to disable the collector, adjust the collection frequency, and enable debugging. It also provides access to things the collector discovered but couldn't liberate. Because the collector augments Python's built-in reference counting, one can turn it off if sure about program won't generate reference cycles.

Intertool: Intertool is a Python package with several functions that work with iterators to create complicated iterators. This module is a memory-efficient, quick tool that can be used alone or in combination to construct iterator algebra. This module implements a set of iterator building pieces based on APL, Haskell, and SML principles. Each has been recast in a Python-friendly format. The module standardizes a core collection of quick, memory-efficient utilities that can be used alone or in tandem. They constitute an "iterator algebra" when combined, making it easy to build specialized tools in pure Python quickly and effectively.

PIL: Pillow is constructed on top of PIL (Python Image Library). PIL is one of Python's most essential image processing modules. Pillow module adds more features, runs on all major operating systems, and has Python 3 support. It can handle a wide range of image formats, including "jpeg," "png," "bmp," "gif," "ppm," and "tiff." With the pillow module, one can do nearly anything with digital photographs. Aside from basic image processing features like point operations, image filtering with built-in convolution kernels, and color space conversions, there's a lot more.

Tqdm: Tqdm is a Python module for displaying clever progress bars that show how far the Python work has progressed. This library can also be used to track the progress of a machine learning model while it is being trained on a large amount of data. Training a machine learning model on a large dataset can take a long time. To check on the status of the model training, one can use the Python Tqdm module to discover how much time is left to train our machine learning model.

Functools: The functools module is for higher-order functions that interact with each other. It includes functions for interacting with other functions and callable objects, allowing one to use or extend them without rewriting them entirely. There are two classes in this module: partial and partial method.

Collection: Different types of containers are available in Python's collection module. A Container is a type of object that may be used to store several objects and provide a mechanism to access and iterate over them. Tuple, List, Dictionary, and other built-in containers are examples.

7. IMPLEMENTATION

7.1 Coding

```
# LOADING AND PREPROCESSING

#Transfer 'jpg' images to an array IMG

def Dataset_loader(DIR, RESIZE, sigmaX=10):

    IMG = []

    read = lambda imname:
np.asarray(Image.open(imname).convert("RGB"))

    for IMAGE_NAME in tqdm(os.listdir(DIR)):

        PATH = os.path.join(DIR, IMAGE_NAME)

        _, ftype = os.path.splitext(PATH)

        if ftype == ".png":

            img = read(PATH)

            img = cv2.resize(img, (RESIZE, RESIZE))

            IMG.append(np.array(img))

    return IMG

benign_train =
np.array(Dataset_loader('/content/drive/MyDrive/BreaKHis
400X/train/benign',224))

malign_train =
np.array(Dataset_loader('/content/drive/MyDrive/BreaKHis
400X/train/malignant',224))

benign_test = np.array(Dataset_loader('/content/drive/MyDrive/BreaKHis
400X/test/benign',224))

malign_test = np.array(Dataset_loader('/content/drive/MyDrive/BreaKHis
400X/test/malignant',224))


# CREATE LABEL

#print label name

label =  {0:"benign", 1:"malignant"}

for i in label.keys() :
```

```
    print(i, label[i])


# breast Cancer: Malignant vs. Benign

# Create labels

benign_train_label = np.zeros(len(benign_train))

malign_train_label = np.ones(len(malign_train))

benign_test_label = np.zeros(len(benign_test))

malign_test_label = np.ones(len(malign_test))

# Merge data

X_train = np.concatenate((benign_train, malign_train), axis = 0)

Y_train = np.concatenate((benign_train_label, malign_train_label),
axis = 0)

X_test = np.concatenate((benign_test, malign_test), axis = 0)

Y_test = np.concatenate((benign_test_label, malign_test_label), axis =
0)

# Shuffle train data

s = np.arange(X_train.shape[0])

np.random.shuffle(s)

X_train = X_train[s]

Y_train = Y_train[s]

# Shuffle test data

s = np.arange(X_test.shape[0])

np.random.shuffle(s)

X_test = X_test[s]

Y_test = Y_test[s]

# To categorical

Y_train = to_categorical(Y_train, num_classes= 2)

Y_test = to_categorical(Y_test, num_classes= 2)
```

 # VISUALIZE DATA

```
#visualize data
```

```
fig, axis = plt.subplots()
axis.bar("test_data", 547, color='b', label='Test Data')
axis.bar("train_data", 1148 , color='r', label='Train Data')
legend = axis.legend()
```

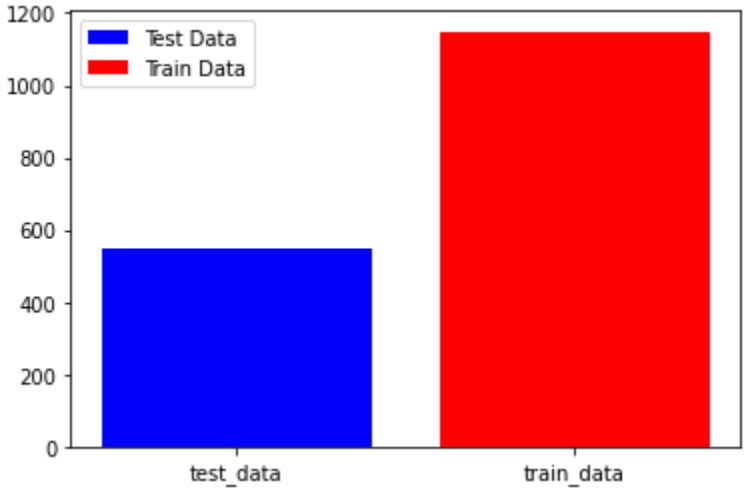

Fig.9

```
#visualize data
fig, axis = plt.subplots()
axis.bar("benign", 547, color='b', label='Benign')
axis.bar("malignant", 1146 , color='r', label='Malignant')
legend = axis.legend()
```

Fig.10

```
train_generator = ImageDataGenerator(
        zoom_range = 2,  # set range for random zoom
        rotation_range = 90,
        horizontal_flip = True,  # randomly flip images
        vertical_flip = True,  # randomly flip images
)
BATCH_SIZE = 16

# TRAIN AND EVALUATION SPLIT
x_train, x_val, y_train, y_val = train_test_split(
    X_train, Y_train,
    test_size = 0.2,
    random_state = 11
)

# DISPLAY IMAGES

# DATA GENERATOR
BATCH_SIZE = 16
# Using original generator
train_generator = ImageDataGenerator(
        zoom_range =2,  # set range for random zoom
        rotation_range = 90,
        horizontal_flip = True,  # randomly flip images
        vertical_flip = True,  # randomly flip images
    )

# BUILDING THE MODEL 1
```

```
#build cnn
def build_model(backbone, lr = 1e-4):
    model = Sequential()
    model.add(backbone)
    model.add(Conv2D(16, (3, 3), padding="valid", activation="relu"))
    model.add(MaxPooling2D(pool_size=(2, 2),strides=(1, 1),
padding="same"))
    model.add(Conv2D(32, (3, 3), padding="valid", activation="relu"))
    model.add(MaxPooling2D(pool_size=(2, 2),strides=(1, 1),
padding="same"))
    model.add(Conv2D(64, (3, 3), padding="valid", activation="relu"))
    model.add(MaxPooling2D(pool_size=(2, 2),strides=(1, 1),
padding="same"))
    model.add(Flatten())
    model.add(layers.Dropout(0.5))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(2, activation='softmax'))


    model.compile(
        loss = 'binary_crossentropy',
        optimizer = Adam(lr=lr),
        metrics = ['accuracy']
    )
    return model
K.clear_session()
gc.collect()
#DenseNet-201 is a convolutional neural network that is 201 layers
deep.
resnet = DenseNet201(
    weights = 'imagenet',
    include_top = False,
    input_shape = (224,224,3)
```

```
)
model = build_model(resnet ,lr = 1e-4)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet201 (Functional)    (None, 7, 7, 1920)        18321984

 conv2d (Conv2D)             (None, 5, 5, 16)          276496

 max_pooling2d (MaxPooling2D  (None, 5, 5, 16)         0
 )

 conv2d_1 (Conv2D)           (None, 3, 3, 32)          4640

 max_pooling2d_1 (MaxPooling  (None, 3, 3, 32)         0
 2D)

 conv2d_2 (Conv2D)           (None, 1, 1, 64)          18496

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 64)         0
 2D)

 flatten (Flatten)           (None, 64)                0

 dropout (Dropout)           (None, 64)                0

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dense (Dense)               (None, 2)                 130

=================================================================
Total params: 18,622,002
Trainable params: 18,392,818
Non-trainable params: 229,184
```

Fig.11

```
#early stop  to avoid overfitting
early_stop =
tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5)


# TRAINING AND EVALUATION FOR MODEL 1
history = model.fit(
    train_generator.flow(x_train, y_train, batch_size = BATCH_SIZE),
    steps_per_epoch = x_train.shape[0] / BATCH_SIZE,
    epochs = 15,
    validation_data = (x_val, y_val),
    callbacks = [early_stop]
)
#evaluate model
model.evaluate(x_val, y_val)


# BUILDING THE MODEL 2
def build_model(backbone, lr=1e-4):
    model = Sequential()
    model.add(backbone)
    model.add(layers.GlobalAveragePooling2D())
    model.add(layers.Dropout(0.5))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(2, activation='softmax'))
    model.compile(
        loss = 'binary_crossentropy',
        optimizer = Adam(lr=lr),
        metrics = ['accuracy']
    )
    return model
```

```
K.clear_session()

gc.collect()

#DenseNet-201 is a convolutional neural network that is 201 layers
deep.

resnet = DenseNet201(

    weights = 'imagenet',

    include_top = False,

    input_shape = (224,224,3)

)

model = build_model(resnet ,lr = 1e-4)

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet201 (Functional)    (None, 7, 7, 1920)        18321984

 global_average_pooling2d (G  (None, 1920)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 1920)              0

 batch_normalization (BatchN  (None, 1920)             7680
 ormalization)

 dense (Dense)               (None, 2)                 3842

=================================================================
Total params: 18,333,506
Trainable params: 18,100,610
Non-trainable params: 232,896
```

Fig.12

```
# Learning Rate Reducer

learn_control = ReduceLROnPlateau(monitor = 'val_acc', patience = 5,
```

```
                                    verbose = 1,factor = 0.2, min_lr =
1e-7)

# Checkpoint

filepath = "weights.best.hdf5"

checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1,
save_best_only=True, mode='max')


# TRAINING AND EVALUATION FOR MODEL 2

history = model.fit(

    train_generator.flow(x_train, y_train, batch_size=BATCH_SIZE),

    steps_per_epoch=x_train.shape[0] / BATCH_SIZE,

    epochs = 30,

    validation_data=(x_val, y_val),

    callbacks=[learn_control, checkpoint]

)
```

7.2 Testing

```
#MODEL 1

#model 1 test 1

img_ = image.load_img("/content/drive/MyDrive/BreaKHis
400X/test/benign/SOB_B_A-14-22549AB-400-004.png", target_size=(244,
244))

imag = image.img_to_array(img_)

imag = np.expand_dims(imag, axis=0)

pred = model.predict(imag)

pred = np.argmax(pred,axis=1)

print(pred)

print(label[pred[0]])

plt.imshow(img_)
```

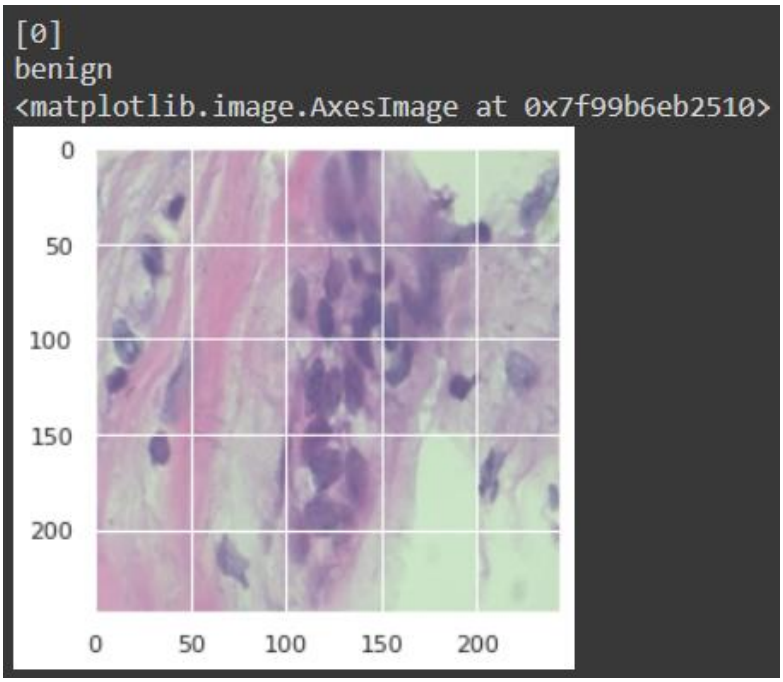

Fig.13

```
#model 1 test 2

img_ = image.load_img("/content/drive/MyDrive/BreaKHis
400X/test/benign/SOB_B_F-14-21998EF-400-011.png", target_size=(244,
244))

imag = image.img_to_array(img_)

imag = np.expand_dims(imag, axis=0)

pred = model.predict(imag)

pred = np.argmax(pred,axis=1)

print(pred)

print(label[pred[0]])

plt.imshow(img_)
```

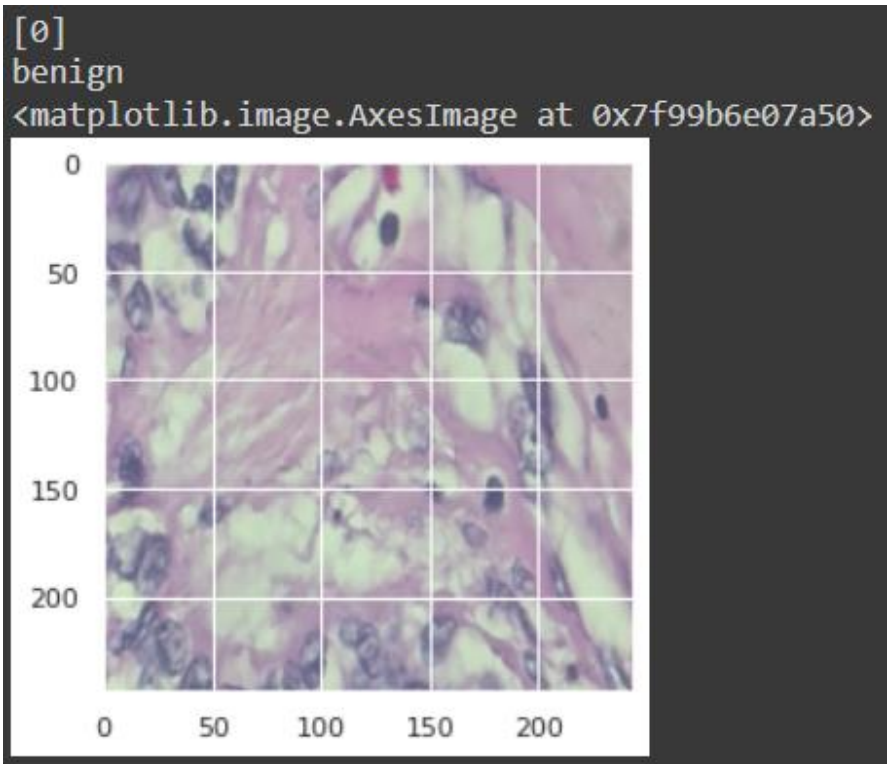

Fig.14

```
#model 1 test 3

img_ = image.load_img("/content/drive/MyDrive/BreaKHis
400X/test/malignant/SOB_M_DC-14-11031-400-008.png", target_size=(244,
244))

imag = image.img_to_array(img_)

imag = np.expand_dims(imag, axis=0)

pred = model.predict(imag)

pred = np.argmax(pred,axis=1)

print(pred)

print(label[pred[0]])

plt.imshow(img_)
```

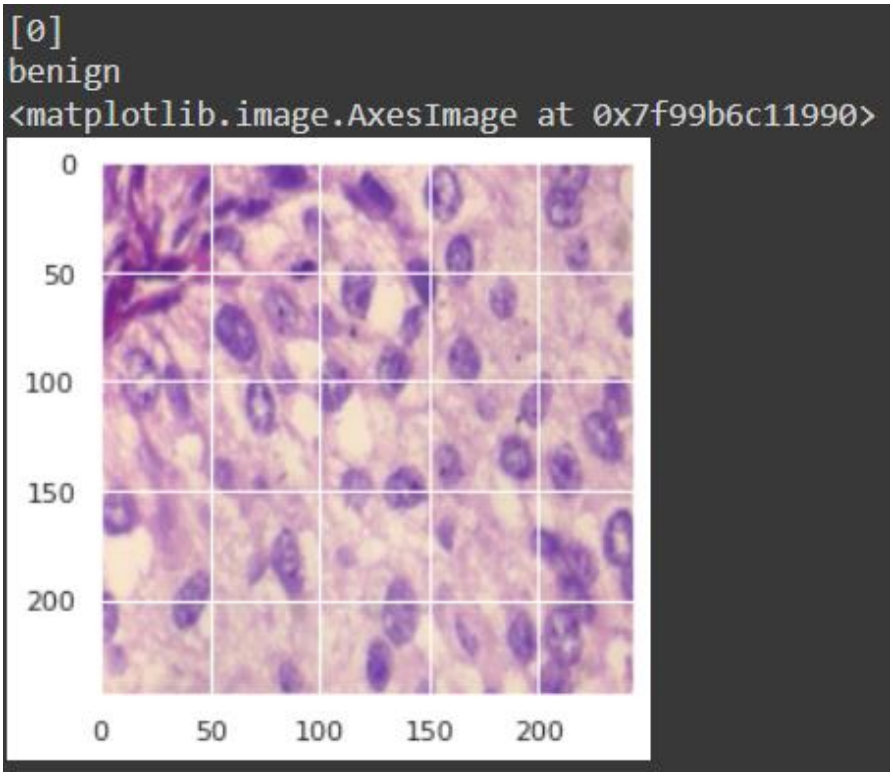

Fig.15

```
#model 1 test 4

img_ = image.load_img("/content/drive/MyDrive/BreaKHis
400X/test/malignant/SOB_M_DC-14-12312-400-026.png", target_size=(244,
244))

imag = image.img_to_array(img_)

imag = np.expand_dims(imag, axis=0)

pred = model.predict(imag)

pred = np.argmax(pred,axis=1)

print(pred)

print(label[pred[0]])

plt.imshow(img_)
```

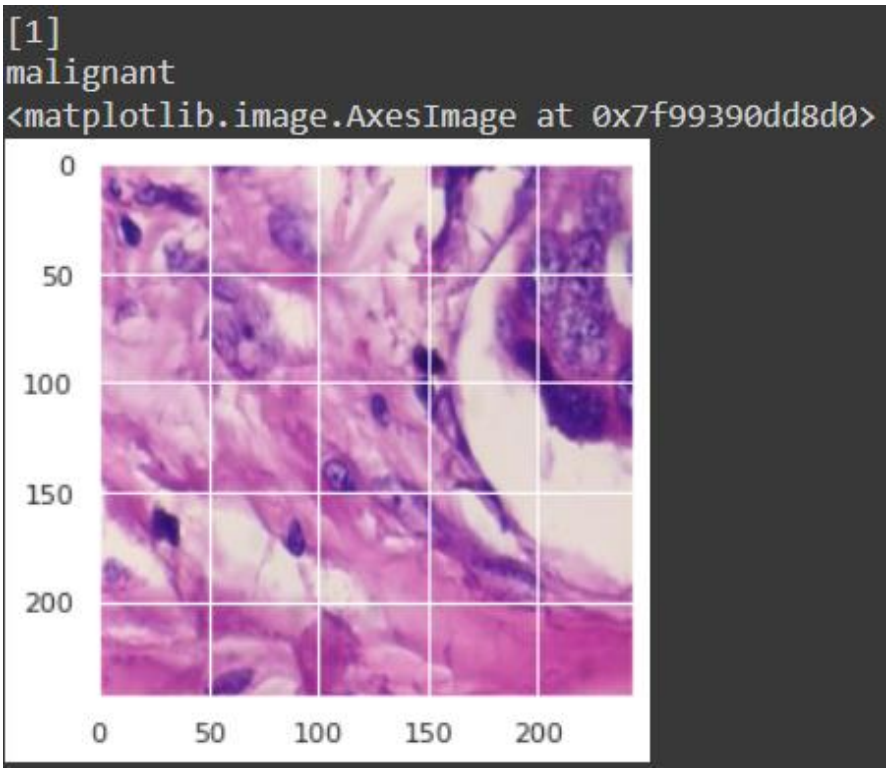

Fig.16

#MODEL 2

```
#model 2 test 1

img_ = image.load_img("/content/drive/MyDrive/BreaKHis
400X/test/benign/SOB_B_A-14-22549AB-400-004.png", target_size=(244,
244))

imag = image.img_to_array(img_)

imag = np.expand_dims(imag, axis=0)

pred = model.predict(imag)

pred = np.argmax(pred,axis=1)

print(pred)

print(label[pred[0]])

plt.imshow(img_)
```

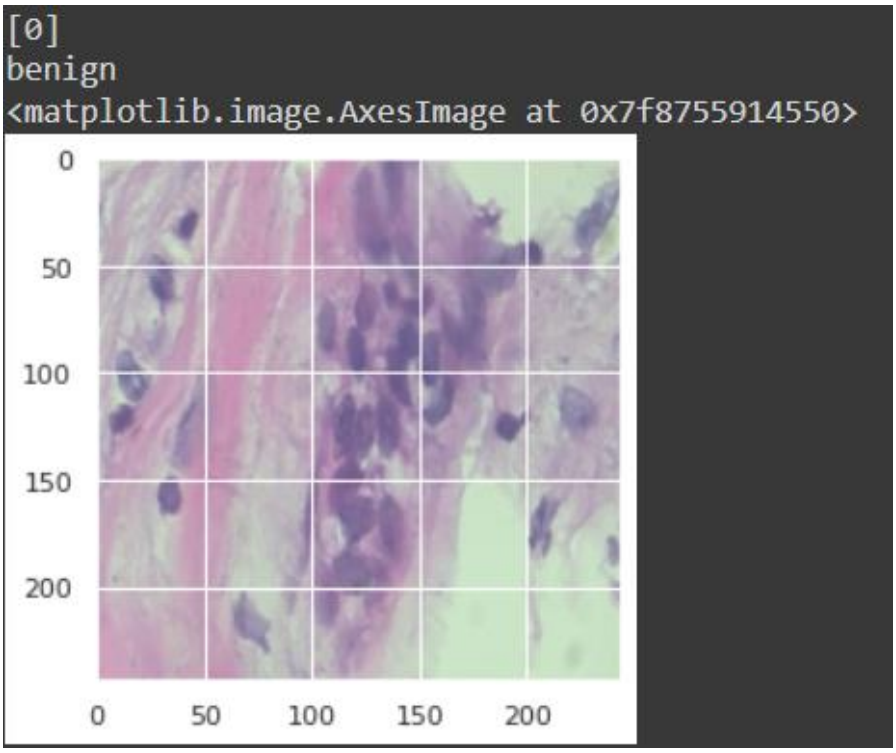

Fig.17

```
#model 2 test 2

img_ = image.load_img("/content/drive/MyDrive/BreaKHis
400X/test/benign/SOB_B_F-14-21998EF-400-011.png", target_size=(244,
244))

imag = image.img_to_array(img_)

imag = np.expand_dims(imag, axis=0)

pred = model.predict(imag)

pred = np.argmax(pred,axis=1)

print(pred)

print(label[pred[0]])

plt.imshow(img_)
```

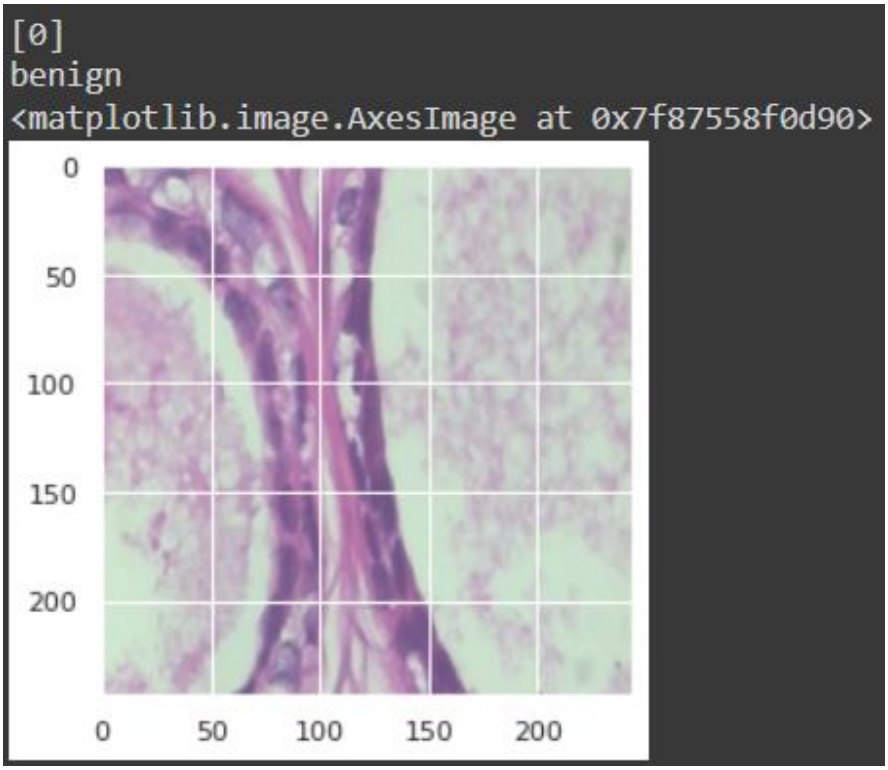

Fig.18

```
#model 2 test 3

img_ = image.load_img("/content/drive/MyDrive/BreaKHis
400X/test/malignant/SOB_M_DC-14-11031-400-008.png", target_size=(244,
244))

imag = image.img_to_array(img_)

imag = np.expand_dims(imag, axis=0)

pred = model.predict(imag)

pred = np.argmax(pred,axis=1)

print(pred)

print(label[pred[0]])

plt.imshow(img_)
```

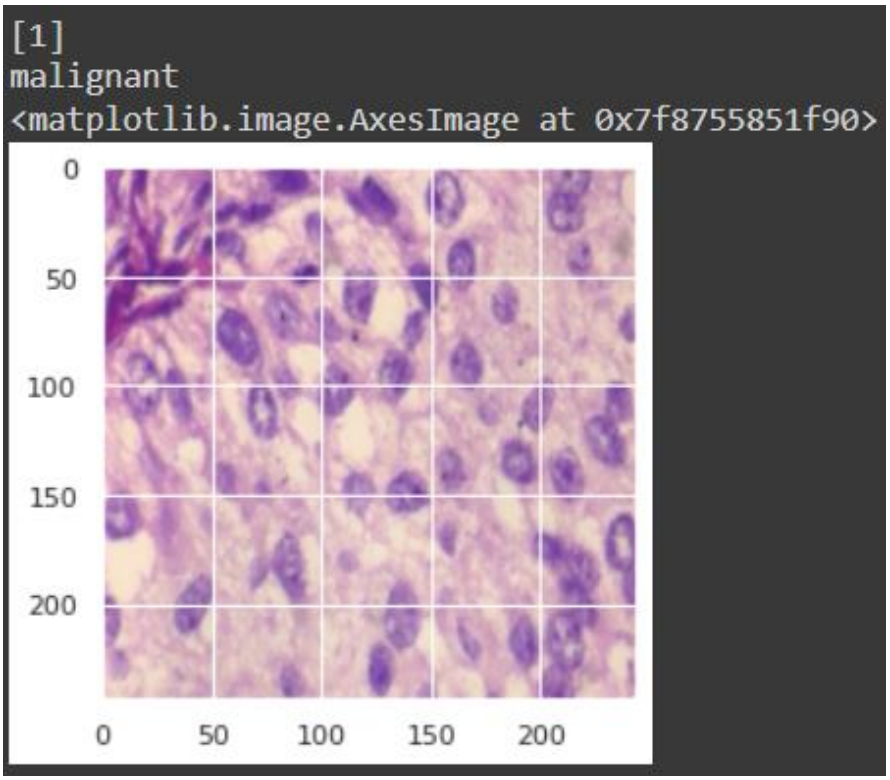

Fig.19

```
#model 2 test 4

img_ = image.load_img("/content/drive/MyDrive/BreaKHis
400X/test/malignant/SOB_M_DC-14-12312-400-026.png", target_size=(244,
244))

imag = image.img_to_array(img_)

imag = np.expand_dims(imag, axis=0)

pred = model.predict(imag)

pred = np.argmax(pred,axis=1)

print(pred)

print(label[pred[0]])

plt.imshow(img_)
```

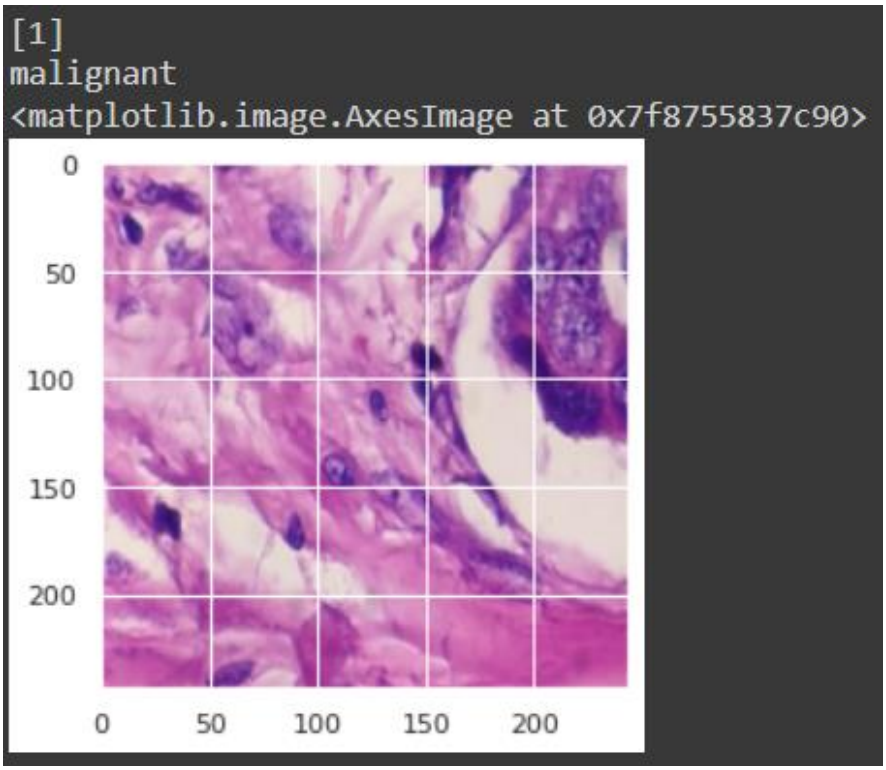

Fig.20

8. RESULTS, ANALYSIS & DISCUSSIONS

CNN has been implemented on the set of 1693 microscopic images of breast tumor tissue collected from 82 patients, 547 of which are benign and 1147 of which are malignant breast cancer tissue images using DenseNet 201 framework using Python v3.6 on Google Colaboratory.

2 CNN models have been employed to determine the best accuracy score in order to detect the cancerous and non-cancerous tissues. Model 1 has an accuracy of 93%, whereas Model 2 has an accuracy of 96%.

8.1 Predicting Breast Cancer Using Machine Learning Classifiers

The below figures represent the classification reports of accuracies of Model 1 and Model 2. The highest level of accuracy is found in Model 2 compared to Model 1.

	precision	recall	f1-score	support
0	0.97	0.73	0.83	176
1	0.89	0.99	0.93	369
accuracy			0.91	545
macro avg	0.93	0.86	0.88	545
weighted avg	0.91	0.91	0.90	545

Fig.21: Classification Report for Model 1

	precision	recall	f1-score	support
0	0.98	0.88	0.93	176
1	0.95	0.99	0.97	369
accuracy			0.96	545
macro avg	0.96	0.94	0.95	545
weighted avg	0.96	0.96	0.96	545

Fig.22: Classification Report for Model 2

8.2 Predicting Breast Cancer Using CNN Model 1

CNN Model 1 has 3 convolution layers with 16, 32, and 64 kernels using MaxPooling 2D.

This model has been trained with 15 epochs, and the batch size is 16. The accuracy score of Model 1 is 93%, as shown in fig 21.

Figs 23 and 24 represent the learning curves' training and validation accuracy and loss.

It can be observed that the values in the model trained and validated have a gap between them when trained with Model 1.

Fig 25 represents the confusion matrix for breast cancer

Fig 26 represents the ROC Curve, with an AUC of 86%, which represents the measure of performance across all possible classification thresholds.

Fig 27 and 28 represent the outputs of the Test Model using CNN Model 1.

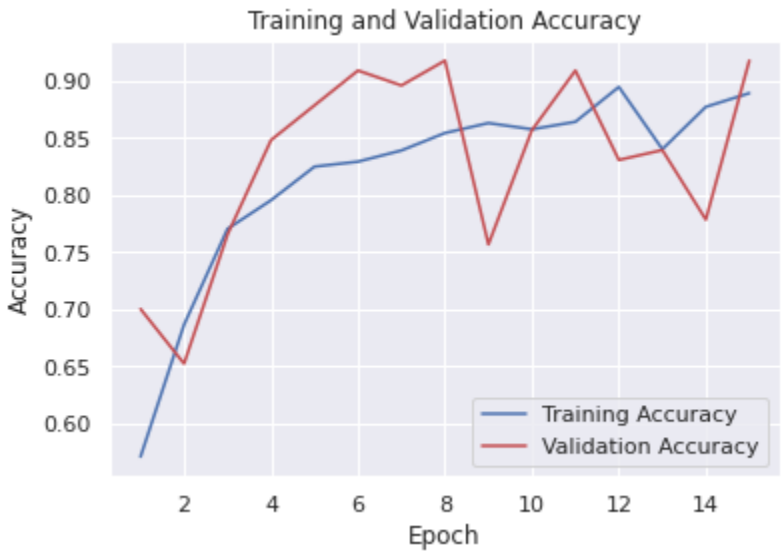

Fig.23: Training and Validation accuracy for Model 1

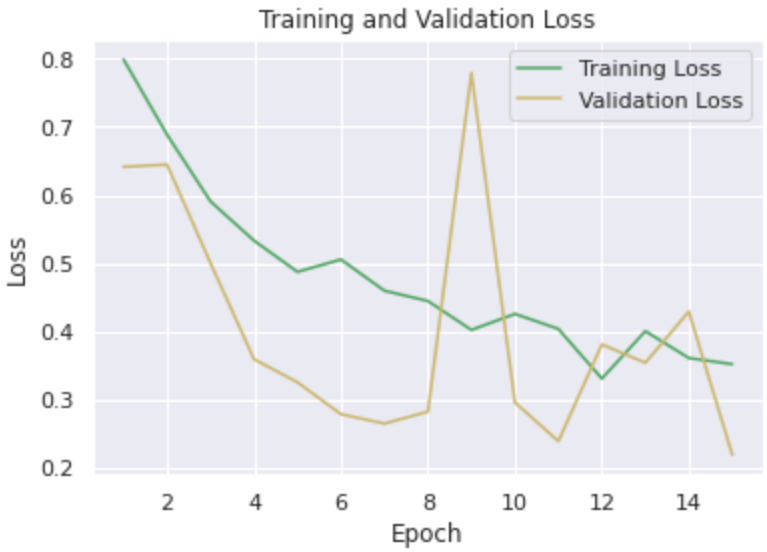

Fig.23: Training and Validation loss for Model 1

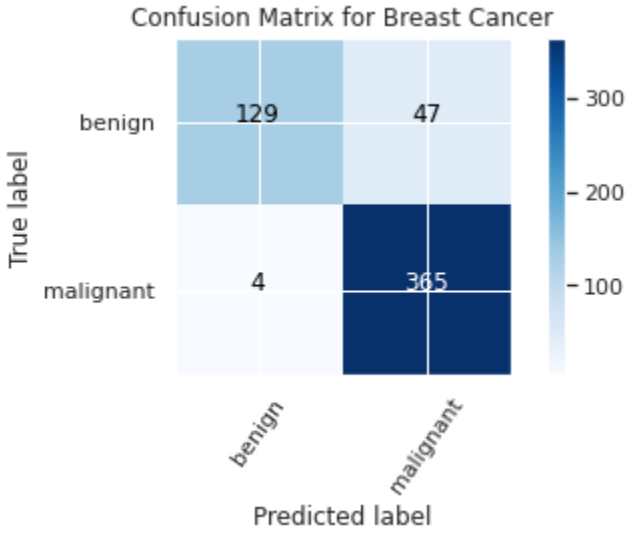

Fig25: Confusion Matrix for Breast Cancer for Model 1

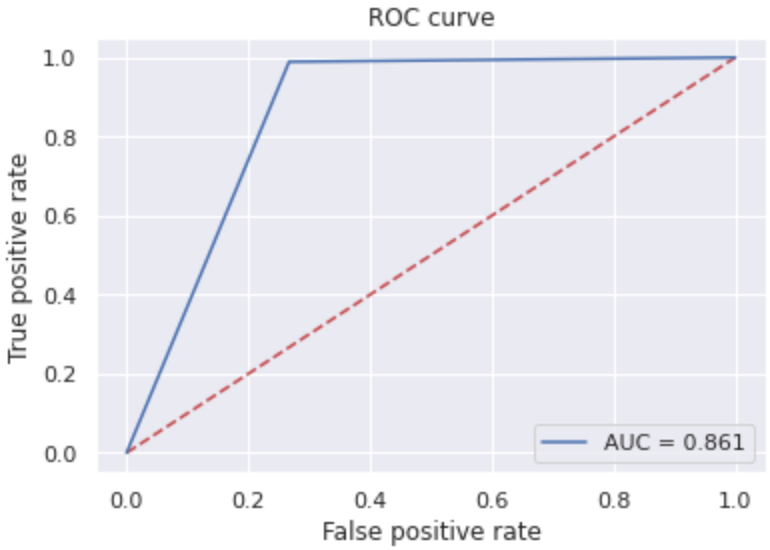

Fig.26: ROC Curve for Model 1

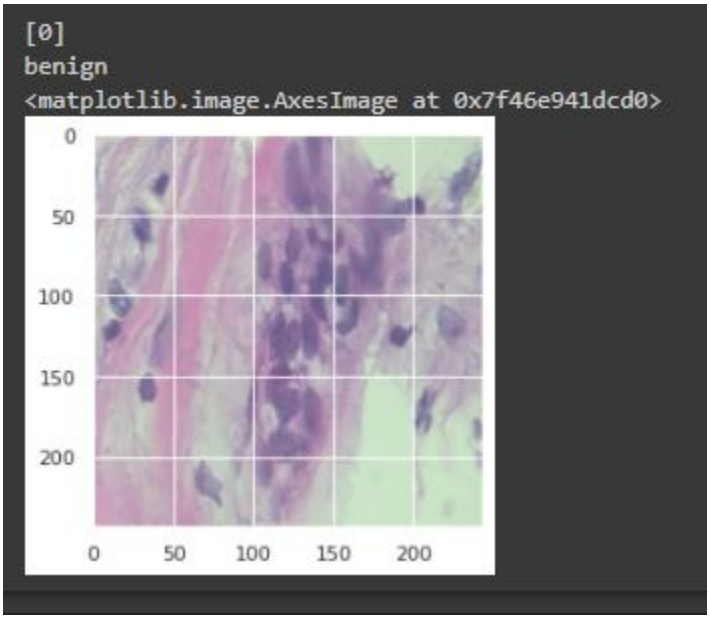

Fig.27: Test Model for Model 1

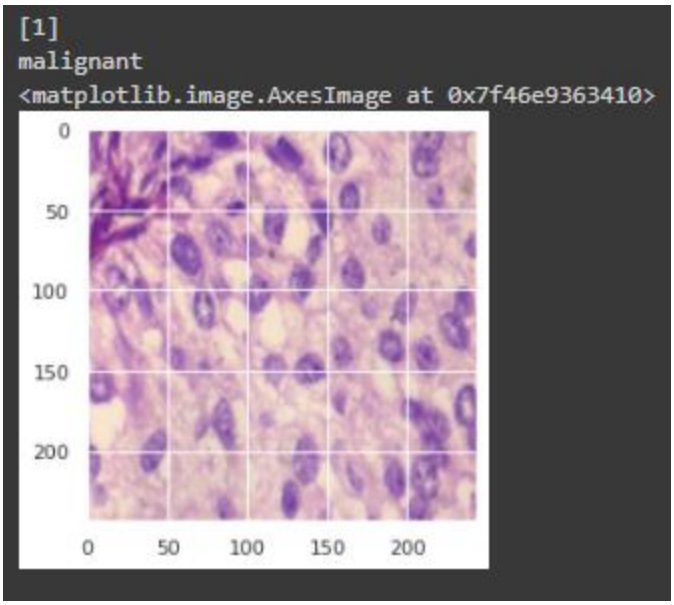

Fig.27: Test Model for Model 1

8.3 Predicting Breast Cancer Using CNN Model 2

CNN Model 2 uses Global Average Pooling. This model has been trained with 30 epochs, and the batch size is 16. CNN Model 2 gets the best result with an accuracy of 96%, as shown in fig 28.

Fig 29 and 30 represent the learning curves' training and validation accuracy and loss.

It can be observed that the values in the model trained and validated have a very narrow gap between them when trained with Model 2, which represents greater accuracy.

Fig 31 represents the confusion matrix for breast cancer for CNN Model 2.

Fig 32 represents the ROC Curve, with an AUC of 93%, representing the measure of performance across all possible classification thresholds.

Fig 33 and 34 represent the outputs of the Test Model using CNN Model 2.

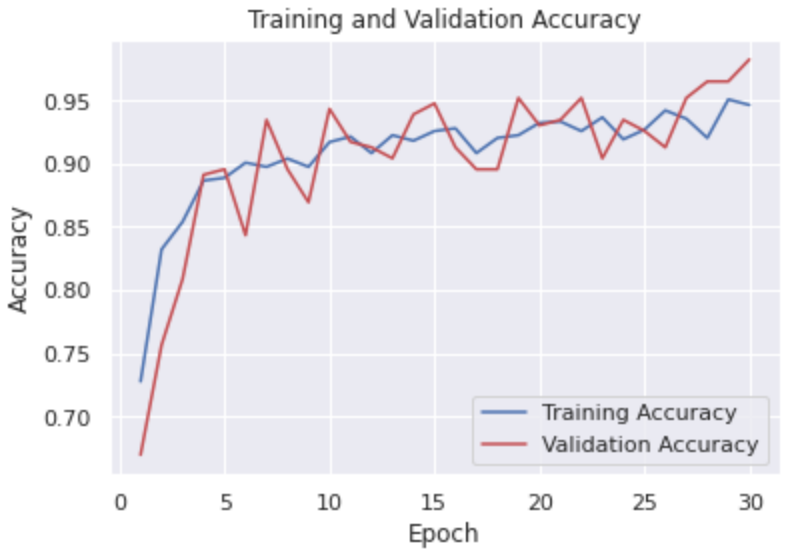

Fig.29: Training and validation accuracy for Model 2

Fig.30: Training and validation loss for Model 2

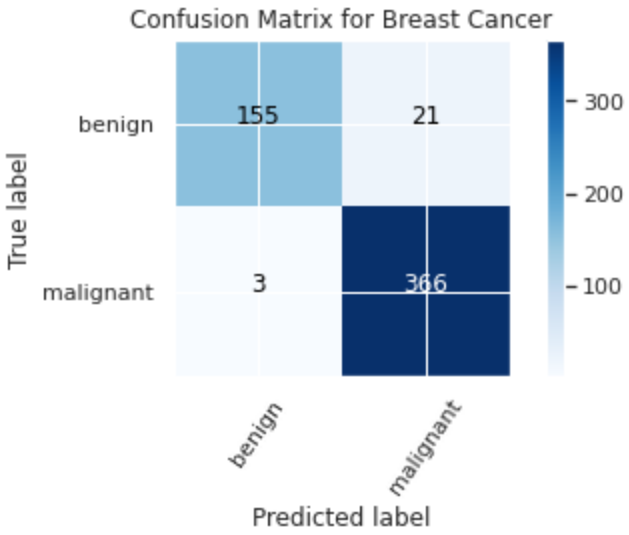

Fig.31: Confusion Matrix for Breast Cancer for Model 2

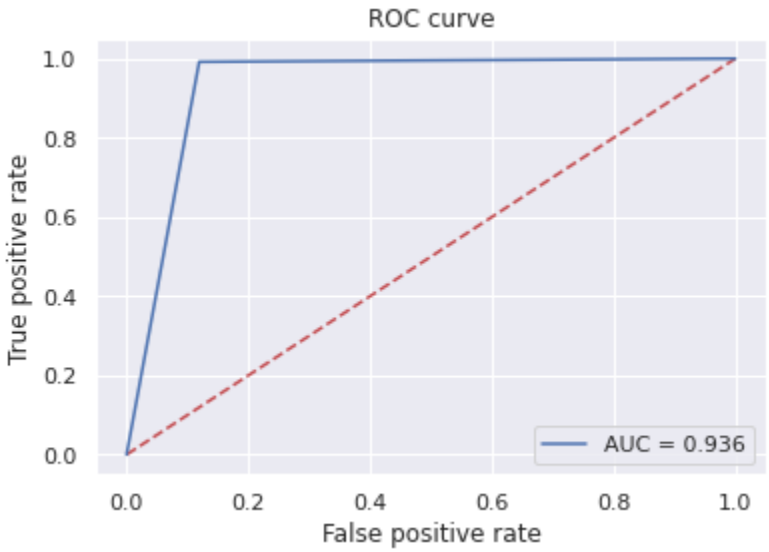

Fig.32: ROC Curve for Model 2

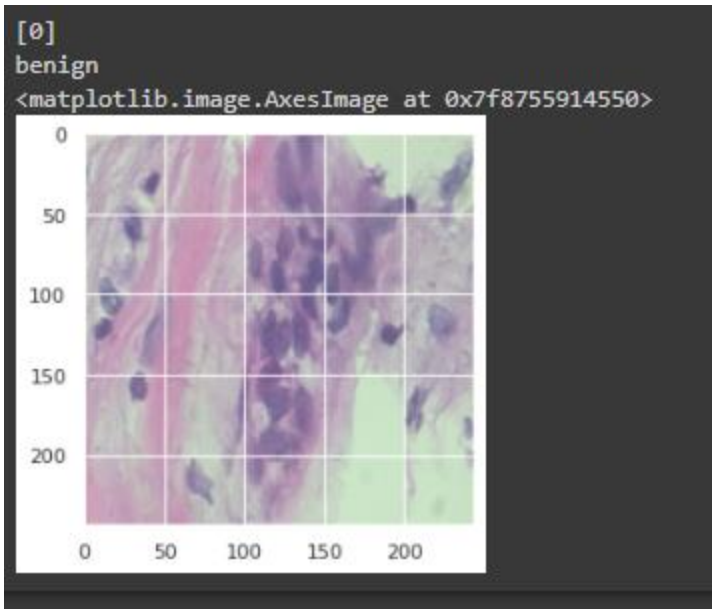

Fig.33: Test Model for Model 2

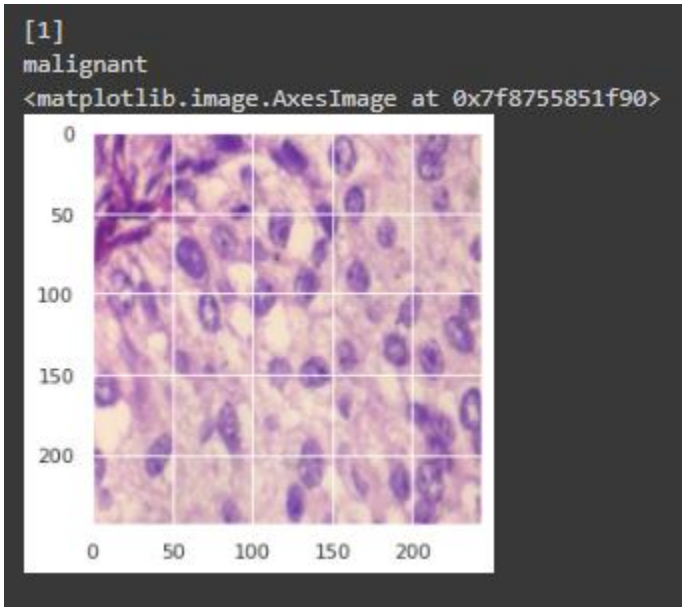

Fig.24: Test Model for Model 2

8.4 Comparison between CNN Model 1 and CNN Model 2

Based on the stated facts, it can be inferred that Model 2 produces superior outcomes than Model 1 when it comes to detecting malignant and benign cells from the dataset of biopsy images of breast cancer tissues.

Parameters	**CNN Model 1**	**CNN Model 2**
Model used	MaxPooling2D	GlobalAveragePooling2D
Epochs	15	30
Accuracy score	91%	98%
Precision	93%	96%
Recall	86%	94%
F1-Score	88%	95%
AUC	86%	93%

9. CONCLUSIONS & FUTURE SCOPE

It's a difficult task to automate breast cancer screening to improve patient care. Above, two distinct CNN architectures were compared for the detection of breast cancer from the dataset of 1693 microscopic images of breast tumor tissue collected from 82 patients, 547 of which are benign and 1147 of which are malignant (700 X 460 pixels, 3-channel RGB, 8-bit depth in each channel, PNG format). The proposed system, which employs CNN Model 2, has a 96% accuracy rate in comparison to CNN Model 1, with an accuracy of 93%. The main scope of this project is for healthcare and oncologists to diagnose cancer accurately as early as possible and to reduce human mistakes in the diagnosis phase. In the future, the usage of AI and ML for efficient diagnosis should be implied to decrease human errors and help people fight cancer as early as possible.

10. REFERENCES

10.1 Research Papers

1. Saad Awadh Alanazi, M. M. Kamruzzaman, Md Nazirul Islam Sarker, Madallah Alruwaili Yousef Alhwaiti, Nasser Alshammari, and Muhammad Hameed Siddiqi "**Boosting Breast Cancer Detection Using Convolutional Neural Network**" Volume 2021 |Article ID 5528622 | https://doi.org/10.1155/2021/5528622

2. Rajesh Kummar, Rajeev Srivastava, Subodh Srivastava, "**Detection and Classification of Cancer from Microscopic Biopsy Images Using Clinically Significant and Biologically Interpretable Features**" Vol. 2015
   View at: Hindawi

3. Philipp Kainz, Michael Pfeiffer, and Martin Urschler, "**Semantic Segmentation of Colon Glands with Deep Convolutional Neural Networks and Total Variation Segmentation**" Vol (revised) 2017 | https://arxiv.org/pdf/1511.06919

4. Bakhan Tofiq Ahmed, "**Lung Cancer Prediction and Detection Using Image Processing Mechanisms: An Overview**" Vol. 1., No. 3, November 2019, pp.20-31

   View at : Research Gate

5. Atsushi Teramoto, Ayumi Yamada, Yuka kiriyama, Tetsuya Tsukamoto, Ke Yan, Ling Zhang, Kazuyoshi Imaizumi, Kuniaki Saito, Hiroshi Fujita "**Automated classification of benign and malignant cells from lung cytological image using deep convolutional neural network**" Vol. 16, 2019.

   View at: Science Direct

10.2 Websites

• Image Feature Extraction https://analyticsindiamag.com/image-feature-extraction-using-scikit-image-a-hands-on-guide/

- Introduction to Image Processing in Python with OpenCV

https://stackabuse.com/introduction-to-image-processing-in-python-with-opencv/

- Basics of Machine Learning Image Classification Techniques

https://iq.opengenus.org/basics-of-machine-learning-image-classification-techniques/

- An Overview on Image Processing Techniques

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1086.6403&rep=rep1&type=pdf

- Database website

https://web.inf.ufpr.br/vri/databases/breast-cancer-histopathological-database-breakhis/