PROJECT REPORT

On

"DETECTION OF BREAST CANCER USING CONVOLUTIONAL NEURAL NETWORK"

Submitted In Partial Fulfilment of the Requirements

For

V SEMESTER

BACHELOR OF COMPUTER APPLICATIONS

SUBMITTED BY

KEERTHANA K R

19mLACB1027

Under the guidance of

Flt.Lt. Harish.H, B.E, MCA, MBA, M.E, (PhD)

Associate Professor

Prof. Bharathi D S B.E, MTech

Assistant Professor

DEPARTMENT OF COMPUTER APPLICATIONS

MAHARANI LAKSHMI AMMANNI COLLEGE FOR WOMEN

Science P.O, Malleshwaram 18th Cross, Bengaluru, Karnataka 560012

MAHARANI LAKSHMI AMMANNI COLLEGE FOR WOMEN

Science P.O, Malleshwaram, Bengaluru, Karnataka 560012

mLAC
ಶೀಲಂ ಪರಮ ಭೂಷಣಂ
Excellence in Education
Since 1972

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that project work entitled **"DETECTION OF BREAST CANCER USING CONVOLUTIONAL NEURAL NETWORK"** *has been successfully submitted by* **KEERTHANA K R** *with bearing register number* **19mLACB1027** *a bonafide student of* **MAHARANI LAKSHMI AMMANNI COLLEGE FOR WOMEN** *in partial fulfilment of the requirement for V semester, BCA prescribed by MLAC Autonomous, BCU during the academic year 2021-2022.*

Flt.Lt. Harish H
HOD,
Department Of Computer Science,
mLAC.

Internal Guides:

1. **Flt. Lt. Harish H**
2. **Prof. Bharathi D S**

EXAMINERS

1.
2.

Declaration

The project **titled "DETECTION OF BREAST CANCER USING CONVOLUTIONAL NEURAL NETWORK"** developed by me in the partial fulfillment of Maharani Lakshmi Ammanni the college. It is a systematic work carried by me under the guidance of **Prof. Bharathi D S** Assistant Professor and Head of Department Flt.Lt.Harish H, Department of Computer Science of Maharani Lakshmi Ammanni College, Bangalore.

I, declare that this same project has not been submitted to any degree or Diploma to the Bangalore University or any other Universities.

Name of the Student:

Date:-

Signature:-

____

ACKNOWLEDGEMENT

We thank our institution **MAHARANI LAKSHMI AMMANNI COLLEGE FOR WOMEN AUTONOMOUS,** for providing all the facilities to complete and furnish the document. We are deeply indebted to the Principal **Dr. A. Shashikala** for allowing us to use all the possible facilities necessary for the documentation.

We also take opportunity to thank our **HOD, Flt. Lt. Harish H,** Department of Computer Science, for his constant guidance and support

We express our sincere gratitude to our guide **Prof. Bharathi D S** without his guidance and supervision the completion of the project would not be possible. We consider ourselves highly indebted for favouring us with his time and undiminished attention.

Finally, we take this opportunity to thank all teaching and non-teaching staffs of **Department of Computer Science of MLACW,** for their immense co-operation and advice during the course of documentation.

Once again we convey our thanks to all those who helped us for the successful completion of this project.

KEERTHANA K R

ABSTRACT

Cancer is the second cause of death in the world. 10 million patients died due to cancer in 2021. Breast cancer is the leading cause of death among women. Breast cancer affects one out of eight females worldwide. It is diagnosed by detecting the malignancy of the cells of breast tissue. Modern medical image processing techniques work on histopathology images captured by a microscope, and then analyse them by using different algorithms and methods. Machine learning algorithms are now being used for processing medical imagery and pathological tools. In deep learning, this is often done by extracting features through a convolutional neural network (CNN) and then classifying employing a fully connected network. In this project, I have used convolution network and obtained prediction accuracy.

CONTENTS

Description

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

The breast is made up of different tissue, ranging from very fatty tissue to very dense tissue. Within this tissue is a network of lobes. Each lobe is made up of tiny, tube-like structures called lobules that contain milk glands. Tiny ducts connect the glands, lobules, and lobes, carrying milk from the lobes to the nipple. The nipple is located in the middle of the areola, which is the darker area that surrounds the nipple. Blood and lymph vessels also run throughout the breast. Blood nourishes the cells. The lymph system drains bodily waste products. The lymph vessels connect to lymph nodes, the small, bean-shaped organs that help fight infection. Groups of lymph nodes are located in different areas throughout the body, such as in the neck, groin, and abdomen. Regional lymph nodes of the breast are those nearby the breast, such as the lymph nodes under the arm.

Cancer begins when healthy cells in the breast change and grow out of control, forming a mass or sheet of cells called a tumor. A tumor can be cancerous or benign. A cancerous tumor is malignant, meaning it can grow and spread to other parts of the body. A benign tumor means the tumor can grow but will not spread.

Breast cancer spreads when the cancer grows into adjacent organs or other parts of the body or when breast cancer cells move to other parts of the body through the blood vessels and/or lymph vessels. This is called a metastasis.

This guide covers both non-invasive (stage 0) as well as early-stage and locally advanced invasive breast cancer, which includes stages I, II, and III. The stage of breast cancer describes how much the cancer has grown, and if or where it has spread.

Although breast cancer most commonly spreads to nearby lymph nodes, it can also spread further through the body to areas such as the bones, lungs, liver, and brain. This is called metastatic or stage IV breast cancer

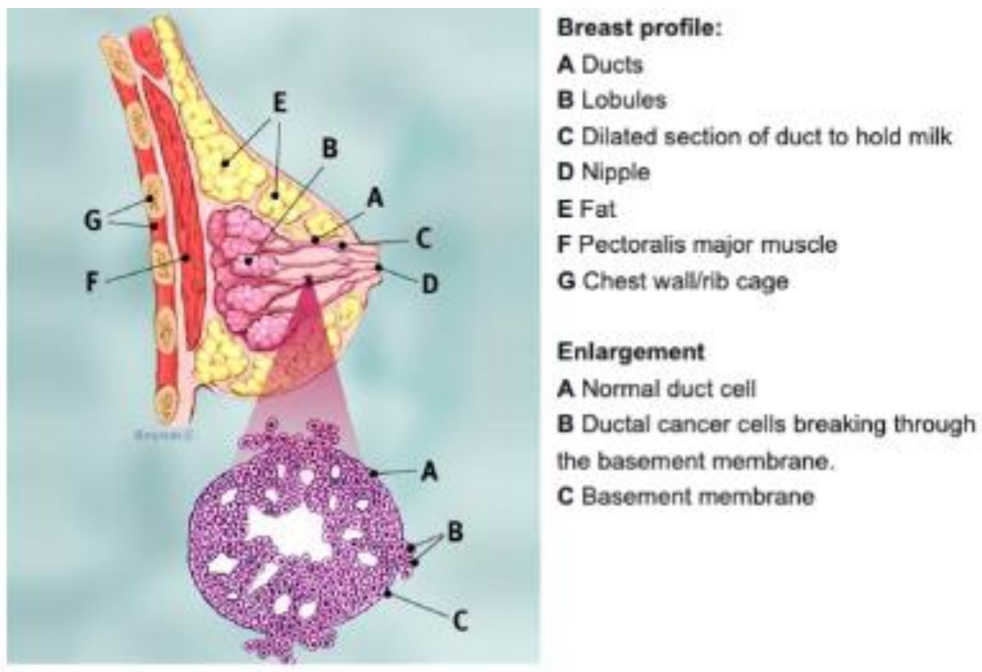

E
B
A
G
C
F
D
A
B
C
Breast profile:
A Ducts
B Lobules
C Dilated section of duct to hold milk
D Nipple
E Fat
F Pectoralis major muscle
G Chest wall/rib cage
Enlargement
A Normal duct cell
B Ductal cancer cells breaking through the basement membrane.
C Basement membrane

Figure 1.1 Invasive Dutual Carcinoma (IDC)

In case of IDC, the cancer would break through the duct cell's membrane and invade or spread to the nearby tissues. In this case the cancer would start in ductal tissue of the breast, duct is the tube that connects the lobules to the nipple. Carcinoma is the type of cancer that starts in the tissue that covers the internal organs, like breast cell in this case. If not detected early, invasive ductal carcinoma can invade the other tissues within breast or even other parts of the body.

Even though invasive ductal carcinoma is most common in older women, it can still affect younger women as well as men. Early detection of the breast cancer will help in increasing the survival rates of the patients. Currently the histopathologists diagnose the tissues extracted from the suspicious tumors and provide information related to type of cancer and its grade when the tumor tests to be malignant. Data science can help in evaluating the suspected tumor tissues through an automated computer vision workflow and provide the diagnosis along with grade there by saving time as well increasing the accuracy of the diagnosis.

1.1 OBJECTIVE

The objective of the whole project is to detect the breast cancer using machine learning algorithms based on image processing.

1.2 PROBLEM STATEMENT

- To understand the challenges associated in identifying the tumors in breast.
- To understand the different image processing methods involved in early detection.
- To understand the effectiveness of CNN algorithm for detecting the breast cancer.

1.3 JUSTIFICATION

Rapid growth in advanced digital technology in the field of breast cancer research as led to wide increase in understanding various clinical conditions, treatment protocols, to increase the survival rate of the patients. Therefore, we have chosen this topic to identify, analyse, and expand the research study for betterment of patient’s wellbeing.

1.4 SCOPE OF WORK

- The purpose of this project is to detect breast cancer using image processing and we have used machine learning algorithm.
- The main symptoms of breast cancer a breast lump or thickening that feels different from the surrounding tissue. Change in the size, shape or appearance of a breast. Changes to the skin over the breast, such as dimpling.

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

Humans have known about breast cancer for a long time. For example, the Edwin Smith Surgical Papyrus describes cases of breast cancer Trusted Source. This medical text dates back to 3,000-2,500 B.C.E.

Our modern approach to breast cancer treatment and research started forming in the 19th century. Consider these milestones:

1882: William Halsted performed the first radical mastectomy. This surgery will remain the standard operation to treat breast cancer until into the 20th century.

1895: The first X-ray is taken. Eventually, low-dose X-rays called mammograms will be used to detect breast cancer.

1898: Marie and Pierre Curie discover the radioactive elements radium and polonium. Shortly after, radium is used in cancer treatment.

1932: A new approach to mastectomy is developed. The surgical procedure is not as disfiguring, and becomes the new standard.

1937: Radiation therapy is used in addition to surgery to spare the breast. After removing the tumor, needles with radium are placed in the breast and near lymph nodes.

1978: Tamoxifen (Nolvadex, Soltamox) is approved by the Food and Drug Administration (FDA) for use in breast cancer treatment. This antiestrogen drug is the first in a new class of drugs called selective estrogen receptor modulators (SERMs).

1984: Researchers discover a new gene in rats. The human version, HER2, is found to be linked with more aggressive breast cancer when overexpressed. Called HER2-positive breast cancer, it isn’t as responsive to treatments.

1985: Researchers discover that women with early-stage breast cancer who were treated with a lumpectomy and radiation have similar survival rates to women treated with only a mastectomy.

1986: Scientists figure out how to clone the HER2 gene.

1995: Scientists can clone the tumor suppressor genes BRCA1 and BRCA2. Inherited mutations in these genes can predict an increased risk of breast cancer.

1996: FDA approves anastrozole (Arimidex) as a treatment for breast cancer. This drug blocks the production of estrogen.

1998: Tamoxifen is found to decrease the risk of developing breast cancer in at-risk women by 50 percentTrusted Source. It’s now approved by the FDA for use as a preventive therapy.

1998: Trastuzumab (Herceptin), a drug targeting cancer cells that are over-producing HER2, is also approved by the FDA.

2006: The SERM drug raloxifene (Evista) is found to reduce breast cancer risk for postmenopausal women who have higher risk. It has a lower chance of serious side effects than tamoxifen.

2011: A large meta-analysisTrusted Source finds that radiation therapy significantly reduces the risk of breast cancer reccurrence and mortality.

2013: The four major subtypesTrusted Source of breast cancer are defined as HR+/HER2 (“luminal A”), HR-/HER2 (“triple negative”), HR+/HER2+ (“luminal B”), and HR-/HER2+ (“HER2-enriched”).

2017: The first biosimilar drug, OgivriTrusted Source (trastuzumab-dkst), is approved by the FDA for breast cancer treatment. Unlike generics, biosimilars are copies of biologic drugs and cost less than branded drugs.

2018: A clinical trial suggests that chemotherapy after surgery doesn’t benefit 70 percent of women with early-stage breast cancer.

2019: EnhertuTrusted Source is approved by the FDA, and this drug proves to be very effective in treating HER2-positive breast cancer that’s metastasized or can’t be removed with surgery.

2020: The drug TrodelvyTrusted Source is approved by the FDA for treating metastatic triple-negative breast cancer for people who haven’t responded to at least two other treatments.

Breast cancer treatments in the 21st century

Breast cancer treatment is becoming more personalized as doctors learn more about the disease.

It’s now seen as a disease with subtypes that have different patterns and ways of acting on the body. The ability to isolate specific genes and classify breast cancer is the beginning of more-tailored treatment options.

For example, the Oncotype DX gene profile test can examine part of a tumor to find out which genes are active in it.

Doctors can determine which patients with early stage breast cancer can be treated with antiestrogen therapy alone, and who would need the addition of chemotherapy

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATIONS

CHAPTER 3
SYSTEM REQUIREMENTS SPECIFICATIONS

3.1 SOFTWARE REQUIREMENTS

- Operating system: Windows 10
- Front end tools: Python, Data Analytics
- Web browser: Google Chrome
- Software: Jupyter Notebook

3.2 HARWARE REQUIREMENTS

- Machine name: HP
- Processor: Intel core i5, 2.50 GHz
- RAM: 4 GB

TECHNOLOGIES USED

3.3 PYTHON

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. It is,

• Free and open-source - You can freely use and distribute Python, even for commercial use.

• Easy to learn - Python has a simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like C++, Java, and C #.

• Portable - You can move Python programs from one platform to another, and run it without any changes.

• Python has a lot of applications. It's used for developing web applications, data science, rapid application development, and so on.

• Python allows you to write programs in fewer lines of code than most of the programming languages.

• The popularity of Python is growing rapidly. Now it's one of the most popular programming languages.

3.4 JUPYTER NOTEBOOK

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

A web application: a browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output. Notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects. Main features of the web application:-

• In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection.

• The ability to execute code from the browser, with the results of computations attached to the code which generated them.

• Displaying the result of computation using rich media representations, such as HTML, LaTeX, PNG, SVG, etc. For example, publication-quality figures rendered by the matplotlib library, can be included inline.

- In-browser editing for rich text using the Markdown markup language, which can provide commentary for the code, is not limited to plain text.

- The ability to easily include mathematical notation within markdown cells using LaTeX, and rendered natively by MathJax.

CHAPTER 4

SYSTEM ANALYSIS

CHAPTER 4
SYSTEM ANALYSIS

4.1 PROPOSED SYSTEM

This is a data analytics project. Our project is based on detection of breast cancer using convolutional neural network.

In this project, we have used Kaggle dataset which is breast histopathology images. The dataset consist of 2,77,524 50X50 RGB images. We have performed image processing on 1,03,303 50X50 RGB images.

After image processing, we have splitted images into 3 categories

i. Testing
ii. Validation
iii. Training

Then, we have developed a model using Convolutional Neural Network (CNN) and we call it as CancerNet.

By performing the model, we have secured the accuracy of 78.28 %

CHAPTER 5

ALGORITHM AND PYTHON LIBRARIES USED

CHAPTER 5

ALGORITHMS

CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks are a type of Deep Learning Algorithm that take the image as an input and learn the various features of the image through filters. This allows them to learn the important objects present in the image, allowing them to discern one image from the other. For example, the convolutional network will learn the specific features of cats that differentiate from the dogs so that when we provide input of cats and dogs, it can easily differentiate between the two.

One important feature of Convolutional Neural Network that sets it apart from other Machine Learning algorithms is its ability to pre-process the data by itself. Thus, you may not spend a lot of resources in data pre-processing. During cold-start, the filters may require hand engineering but with progress in training, they are able to adapt to the learned features and develop filters of their own. Therefore, CNN is continuously evolving with growth in the data.

The architecture of the Convolutional Neural Network is as follows

- **INPUT –** As discussed above, a typical image in the CIFAR 10 data will hold images if dimensions 32x32x3 where the depth denotes the number of channels (RGB) in the image.

- **CONV** layer is responsible for computing the dot product between the weights of the neuron and the region of the input image to which share a connection. Here, the dimensions become 32x32x12 denoting the 12 filters which the neural network makes use of.
- The third layer consists of **RELU** which is for the application of the activation function to our resultant dot product. The size of this result does not change.

- The fourth **POOL** layer will down sample the spatial dimensions of the image i.e. width and height. This will reduce the dimensions to 16x16x12.

- The fully connected layer will be responsible for computing the class score, leading to a final volume of 1x1x10. The 10 here represent the categories of CIFAR-10.

PYTHON LIBRARIES

Keras

Keras is an open- source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the Tensorflow library.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. Keras has support for convolutional neural networks.

Tensorflow

Tensor Flow is a free and open- source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

Tensor Flow serve as the core platform and library for machine learning. Tensor Flow's API'S use Keras to allow users to make their own machine learning models. In addition to build and to train our model, Tensor Flow can also help load the data to train the model

NumPy

It is a general-purpose array-processing package. This package provides a high-performance multidimensional array object, and tools to work on with this array.

Matplotlib

Matrix plot Library in python is mainly used for data visualization. It is used for making 2D plots from data in arrays.

OS

This module in python functions for creating and removing a directory(folder), fetching its contents, changing and identifying the current directory etc.

Sklearn

Scikit-learn is the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

Skimage

Scikit-image is a Python package dedicated to image processing, and using natively NumPy arrays as image objects.

CHAPTER 6

IMPLEMENTATION

CHAPTER 7

IMPLEMENTATION

METHODOLOGY

6.1 WORKFLOW OF THE PROJECT

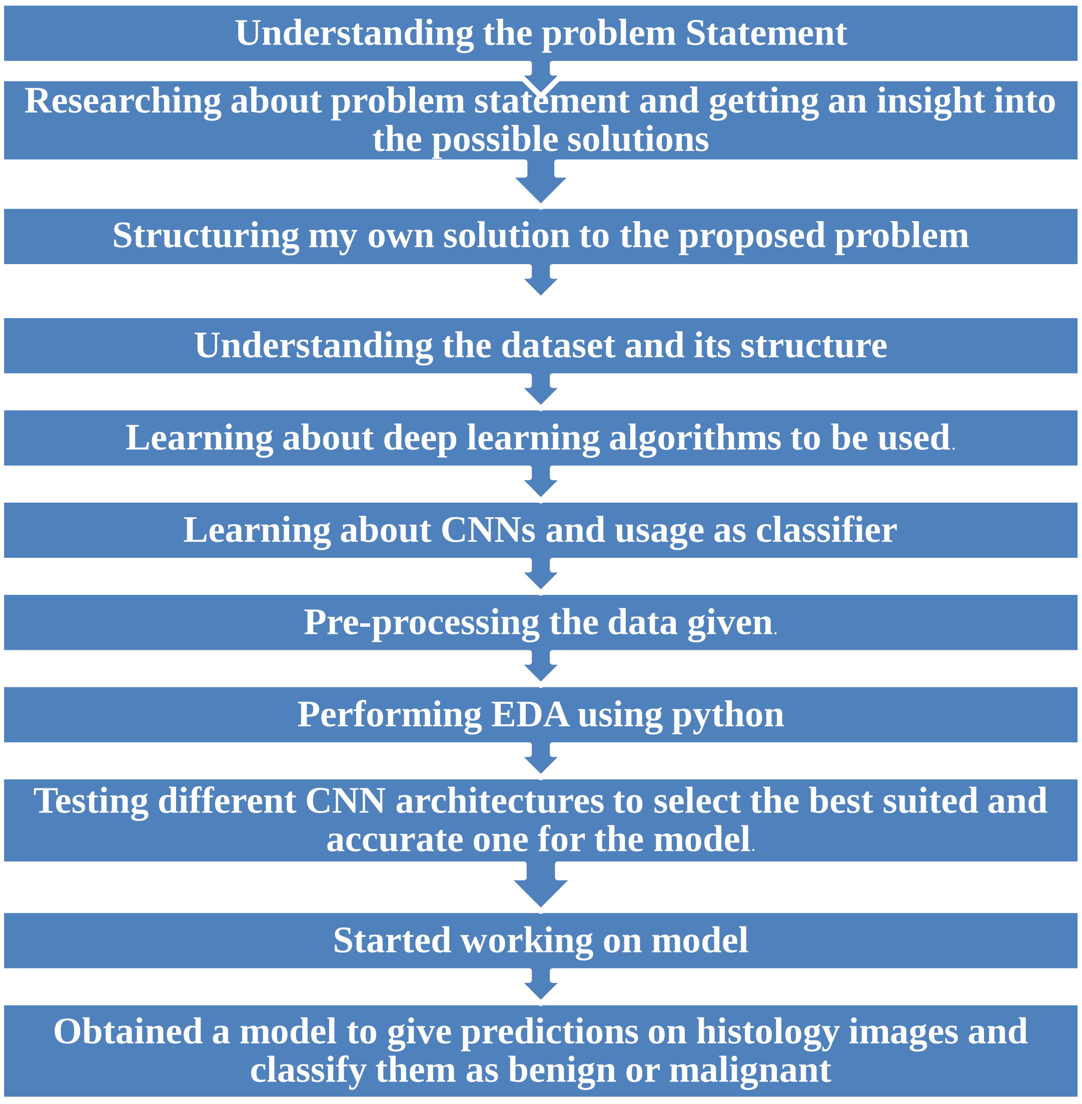

Understanding the problem Statement
Researching about problem statement and getting an insight into the possible solutions
Structuring my own solution to the proposed problem
Understanding the dataset and its structure
Learning about deep learning algorithms to be used.
Learning about CNNs and usage as classifier
Pre-processing the data given.
Performing EDA using python
Testing different CNN architectures to select the best suited and accurate one for the model.
Started working on model
Obtained a model to give predictions on histology images and classify them as benign or malignant

Figure 6.1 Workflow of the project

IMPLEMENTATION

6.2 DATASET USED

The dataset consist of 2,77,524 50X50 pixel RGB digital image patches that were derived from 162 H&E-stained breast histopathology samples. These images are small patches that were extracted from digital images of breast tissue samples. The breast tissue contains many cells but only some of them are cancerous. Patches that are labelled “0” are benign (non-cancerous) cells and labelled “1” are malignant (cancerous) cells.

6.3 IMAGE PREPROCESSING

Most of the pixels in the image are redundant and do not contribute substantially to the intrinsic information of an image. This can be achieved by compression techniques. We begin the implementation by processing the images in the dataset. This is achieved with the help of the keras and tensorflow libraries in Python. There are many other modules that can be used in this step e.g. MATLAB or other image processing libraries or software. This is necessary to remove redundancy from the input data which only contributes to the computational complexity of the network without providing any significant improvements in the result. The aspect ratio of the original slide is preserved since both the dimensions are reduced by a factor of 2, giving an image which is 1/4th in area that is of dimension 50×50 pixels.

6.4 FEATURE EXTRACTION

For features extraction, we use pre-trained ResNet-50, InceptionV3 and VGG-16 networks. We remove fully connected layers from each model to allow the networks to consume images of an arbitrary size. In ResNet-50 and InceptionV3, we convert the last convolutional layer consisting of 2048 channels via GlobalAveragePooling into a one-dimensional feature vector with a length of 2048. With VGG-16 we apply the GlobalAveragePooling operation to the four internal convolutional layers: block2, block3, block4, block5 with 128, 256, 512, 512 channels respectively.

6.5 CLASSIFICATION

In this project, we will build a classifier to train on 80% of a breast cancer histology image dataset. We will keep 10% of the data for validation. Using Keras, we will define a CNN (Convolutional Neural Network) and called it as CancerNet and train it on our images. We'll then derive a confusion matrix to analyse the performance of the model.

6.6 DATASET SPLIT

We have splitted the dataset into 3 categories

1. Training set

Negative images(0)

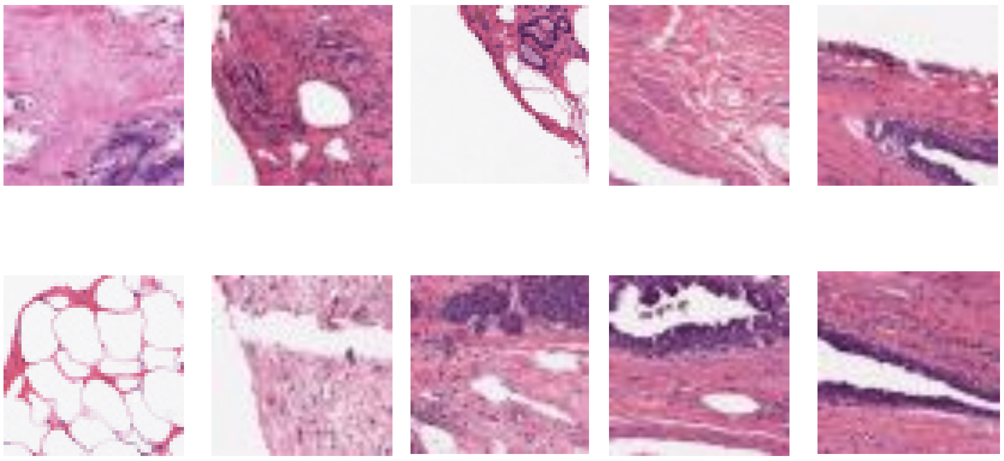


Figure 6.2 Training set negative images

Positive images(1)

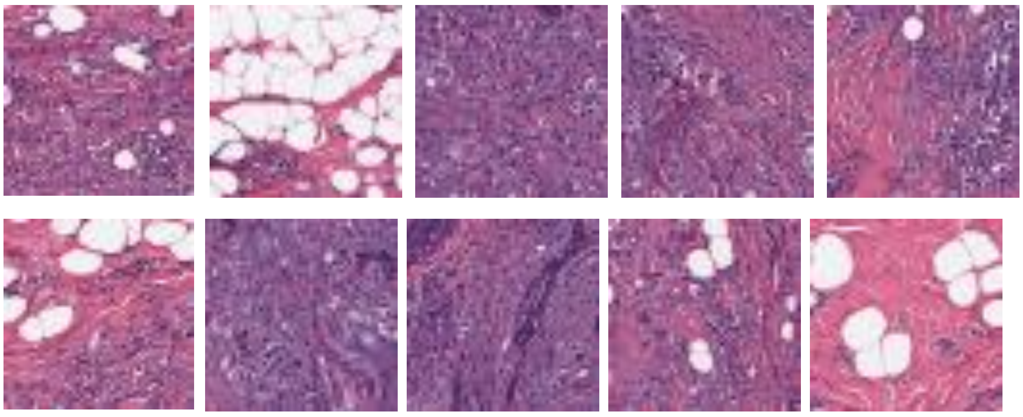


Figure 6.3 Training set positive images

2. Validation set

Negative images(0)

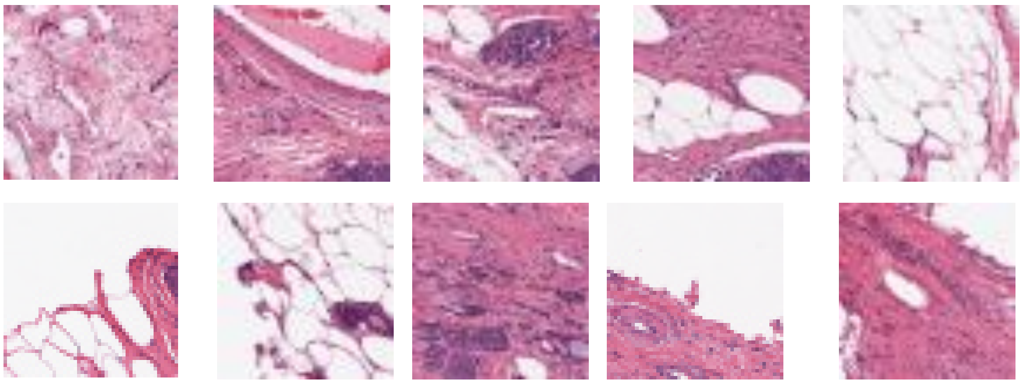


Figure 6.4 Validation set negative images

Positive images(1)

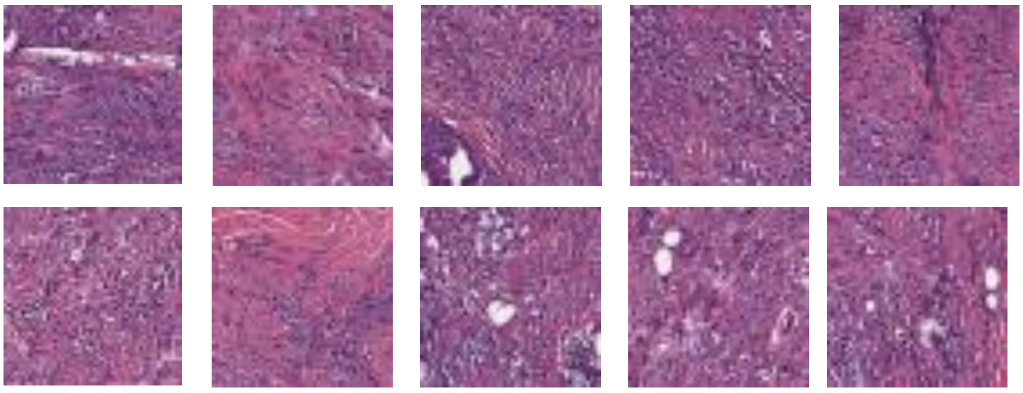


Fig 6.5 Validation set positive images

3. Testing

Negative images(0)

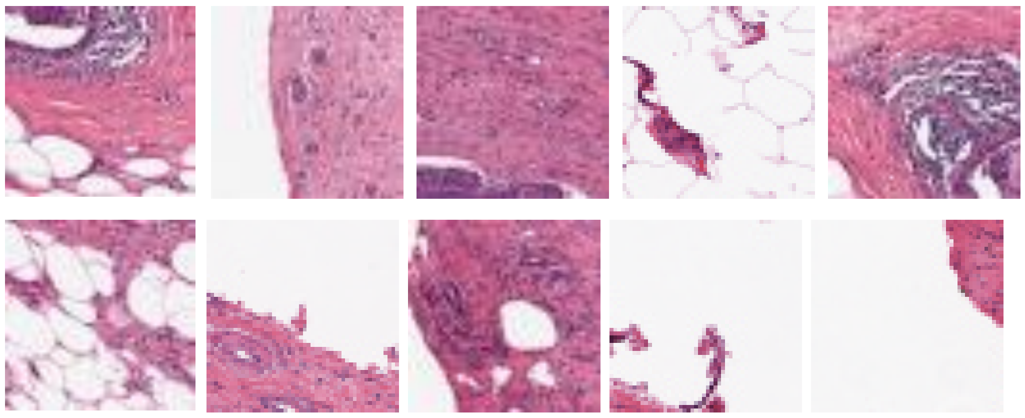


Figure 6.6 Testing set negative images

Positive images (1)

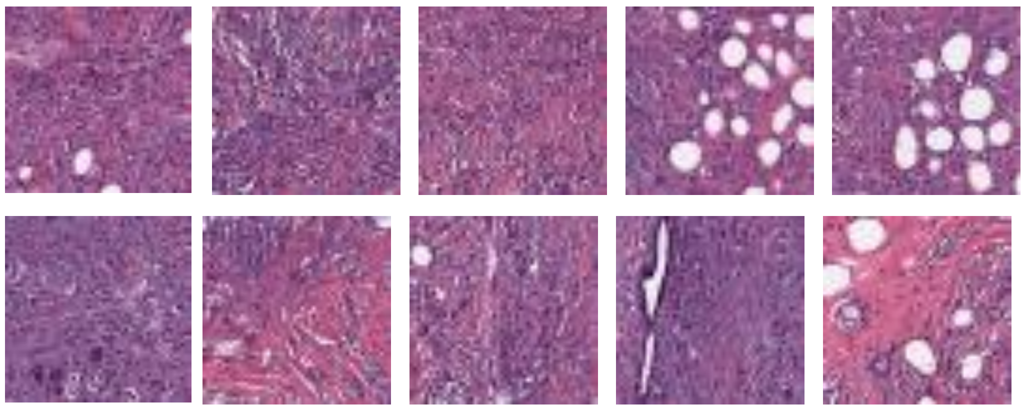


Figure 6.7 Testing set positive images

6.7 CANCERNET

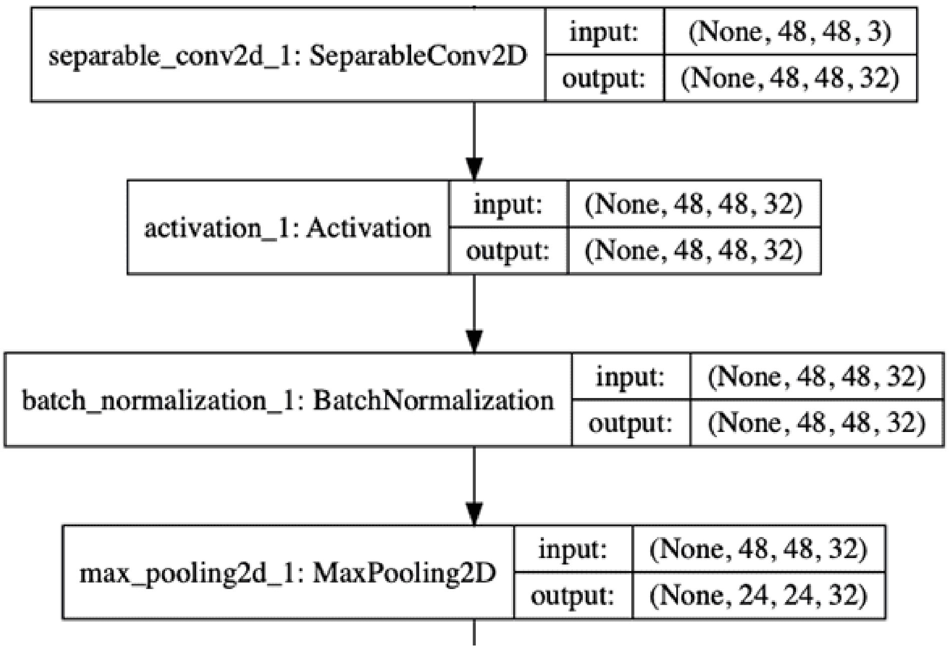

separable_conv2d_1: SeparableConv2D
input: (None, 48, 48, 3)
output: (None, 48, 48, 32)
activation_1: Activation
input: (None, 48, 48, 32)
output: (None, 48, 48, 32)
batch_normalization_1: BatchNormalization
input: (None, 48, 48, 32)
output: (None, 48, 48, 32)
max_pooling2d_1: MaxPooling2D
input: (None, 48, 48, 32)
output: (None, 24, 24, 32)

Figure 6.8 Classification architecture

To implement the architecture we used the Keras deep learning library and designed a network appropriately named "CancerNet" which:

1. Uses exclusively *3×3* CONV filters, similar to VGGNet
2. Stacks multiple *3×3* CONV filters on top of each other prior to performing max-pooling (again, similar to VGGNet)
3. But unlike VGGNet, uses depth wise separable convolution rather than standard convolution layers

CHAPTER 7

CODING

CHAPTER 7
CODING

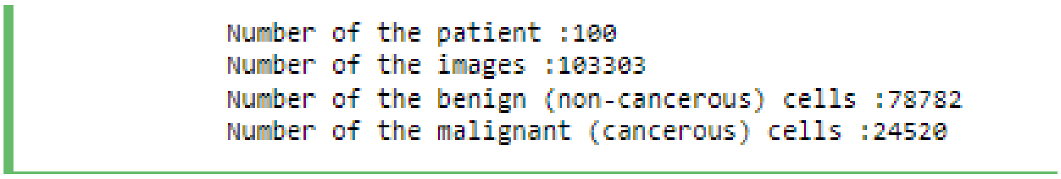
In[1]:

```
import os , glob
from imutils import paths
# linear algebra
import numpy as np
# data processing, CSV file I/O (e.g. pd.read_csv)
import pandas as pd import matplotlib.pyplot as plt
import seaborn as sns
import random
import shutil
import subprocess, sys

import skimage
from skimage.io import imread , imread_collection

from sklearn.metrics import accuracy_score , classification_report
from sklearn.preprocessing import LabelEncoder,StandardScaler

import keras
from keras.applications import mobilenet
from keras.layers import Dense ,Dropout
from keras.models import Sequential
```

In[2]:

```
# Get some high level stats from the images
images_path ='../PROJECT/dataset/*'
patient_list = list(glob.glob(images_path))
print(f'Number of the patient :{len(patient_list)}')

images_count = 0
for i in range(0 , len(patient_list)):
    images_count+=len(glob.glob(patient_list[i]+'/*/*'))
print(f'Number of the images :{images_count}')

benign_count = 0
for i in range(0 , len(patient_list)):
    benign_count+=len(glob.glob(patient_list[i]+'/0/*'))
print(f'Number of the benign (non-cancerous) cells
:{benign_count}')

malignant_count = 0
for i in range(0 , len(patient_list)):
    malignant_count+=len(glob.glob(patient_list[i]+'/1/*'))
print(f'Number of the malignant (cancerous) cells
:{malignant_count}')
```

Out[2]:

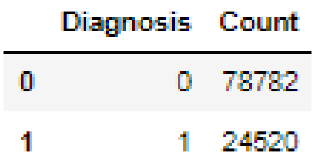


```
Number of the patient :100
Number of the images :103303
Number of the benign (non-cancerous) cells :78782
Number of the malignant (cancerous) cells :24520
```

In[3]:

```
# Build the Dataframe with counts
BC_list = [["0", benign_count], ["1", malignant_count]]
BC_df = pd.DataFrame(BC_list, columns=['Diagnosis', 'Count'])
BC_df.head()
```

Out[3]:

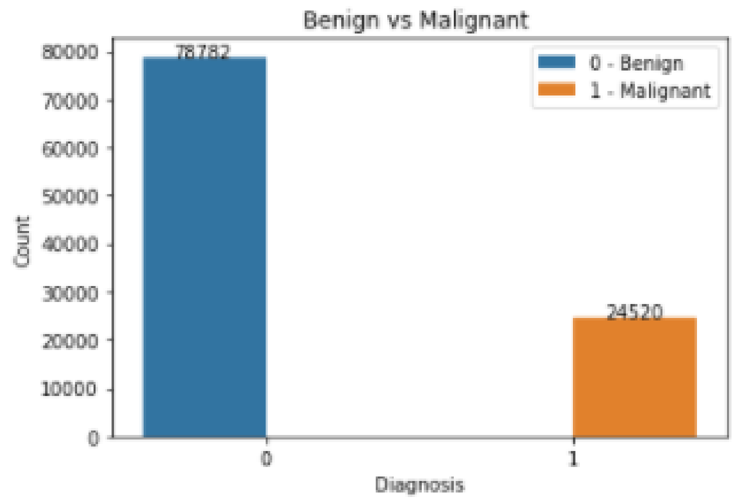


	Diagnosis	Count
0	0	78782
1	1	24520

In[4]:

```
# Plot the bar chart
ax = sns.barplot(x="Diagnosis", y="Count", data=BC_df,
hue="Diagnosis")
plt.title('Benign vs Malignant ')
for p in ax.patches:
    ax.annotate('{:.0f}'.format(p.get_height()),          (p.get_x()+0.1,
p.get_height()+10))

h, l = ax.get_legend_handles_labels()
ax.legend(h, ['0 - Benign', '1 - Malignant'], loc='best')
plt.show()
print(f'\nThe  difference  between  benign(0)  and  malignant(1)  :
{(benign_count-malignant_count)}')
```

Out[4]:



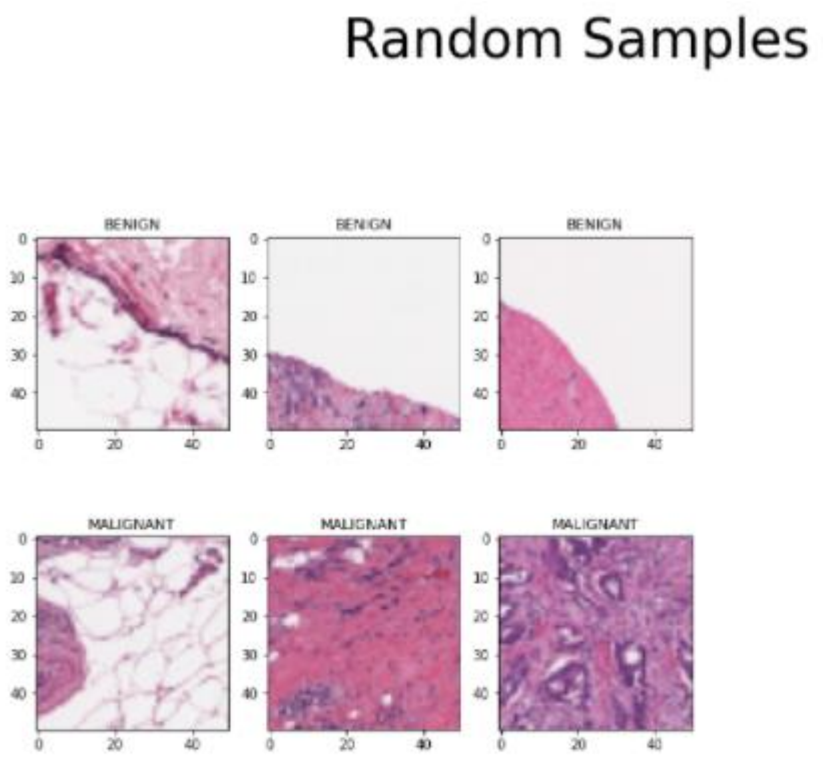
```
The difference between benign(0) and malignant(1) : 54262
```

In[5]:

```
# Plot random cell samples
# Create list of patient objects
test=[glob.glob(patient_list[i]+'/*/*')           for           i           in
range(0,len(patient_list))]
```

```
fig,axes = plt.subplots(figsize=(15,20))



# Get the random sample prepared for 3 patients
ran_sample = [random.choice(test) for x in range(0,3)]

# Loop through and plot first (to get benign(0)) and last (to get
mailgnant(1)) images.
for i in range(0,len(ran_sample)):
    # Plot first image
    path=ran_sample[i][0]
    img = imread(path)
    axes =plt.subplot(5,5,i+1)
    plt.imshow(img)
    axes.set_title('BENIGN',fontsize=11)



    # Plot last image
    path=ran_sample[i][len(ran_sample[i])-1]
    img = imread(path)
    axes =plt.subplot(5,5,i+6)
    plt.imshow(img)
    axes.set_title('MALIGNANT',fontsize=11)

fig.suptitle('Random Samples',fontsize=40)
plt.show()
```

Out[5]:



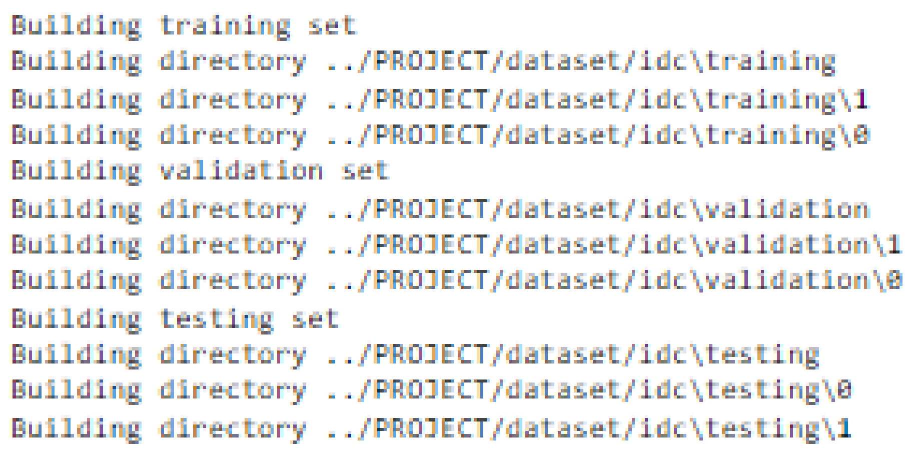
In[6]:

```
# Set generic vairbales
# Set parameters for splitting the input data
images_path ='../PROJECT/dataset'
BASE_PATH = "../PROJECT/dataset/idc"
TRAIN_PATH = os.path.sep.join([BASE_PATH, "training"])
VAL_PATH = os.path.sep.join([BASE_PATH, "validation"])
TEST_PATH = os.path.sep.join([BASE_PATH, "testing"])
TRAIN_SPLIT = 0.8
VAL_SPLIT = 0.2
```

In[7]:

```
# Below section copies the images from source to the respective directories for preparing
# testing, training, and validation datasets.

# Split the data now
originalPaths=list(paths.list_images(images_path))
random.seed(7)
random.shuffle(originalPaths)
index=int(len(originalPaths)*TRAIN_SPLIT)
trainPaths=originalPaths[:index]
testPaths=originalPaths[index:]
index=int(len(trainPaths)*VAL_SPLIT)
valPaths=trainPaths[:index]
trainPaths=trainPaths[index:]
datasets=[("training", trainPaths, TRAIN_PATH),
          ("validation", valPaths, VAL_PATH),
          ("testing", testPaths, TEST_PATH)
]
for (setType, originalPaths, basePath) in datasets:
    print(f'Building {setType} set')
    if not os.path.exists(basePath):
        print(f'Building directory {basePath}')
        os.makedirs(basePath)

    for path in originalPaths:
        file=path.split(os.path.sep)[-1]
        label=file[-5:-4]
        labelPath=os.path.sep.join([basePath,label])
        if not os.path.exists(labelPath):
            print(f'Building directory {labelPath}')
            os.makedirs(labelPath)
        newPath=os.path.sep.join([labelPath, file])
        shutil.copy2(path, newPath)
```

Out[7]:



```
Building training set
Building directory ../PROJECT/dataset/idc\training
Building directory ../PROJECT/dataset/idc\training\1
Building directory ../PROJECT/dataset/idc\training\0
Building validation set
Building directory ../PROJECT/dataset/idc\validation
Building directory ../PROJECT/dataset/idc\validation\1
Building directory ../PROJECT/dataset/idc\validation\0
Building testing set
Building directory ../PROJECT/dataset/idc\testing
Building directory ../PROJECT/dataset/idc\testing\0
Building directory ../PROJECT/dataset/idc\testing\1
```

In[8]:

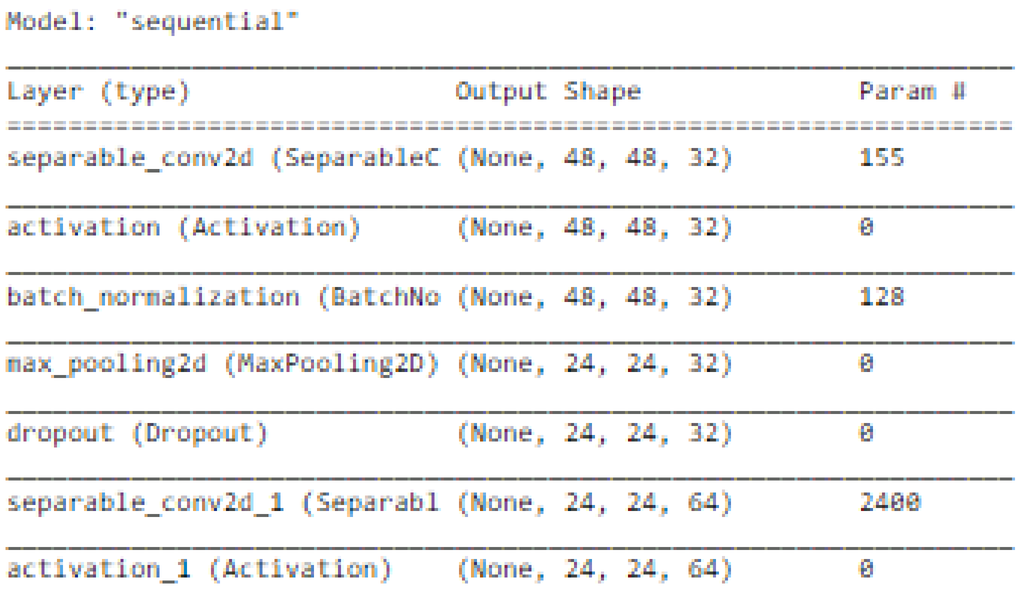
```
from keras.models import sequential
from keras.layers import BatchNormalization
from keras.layers.convolutional import SeparableConv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dropout
from keras.layers.core import Dense
from keras import backend as K
```

```
height=48
width=48
depth=3
classes=2

In[9]:

# Build model
model=Sequential()
shape=(height,width,depth)
channelDim=-1
if K.image_data_format()=="channels_first":
    shape=(depth,height,width)
    channelDim=1
model.add(SeparableConv2D(32,(3,3),
padding="same",input_shape=shape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=channelDim))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(SeparableConv2D(64, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=channelDim))
model.add(SeparableConv2D(64, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=channelDim))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(SeparableConv2D(128, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=channelDim))
model.add(SeparableConv2D(128, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=channelDim))
model.add(SeparableConv2D(128, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=channelDim))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(classes))
model.add(Activation("softmax"))
```

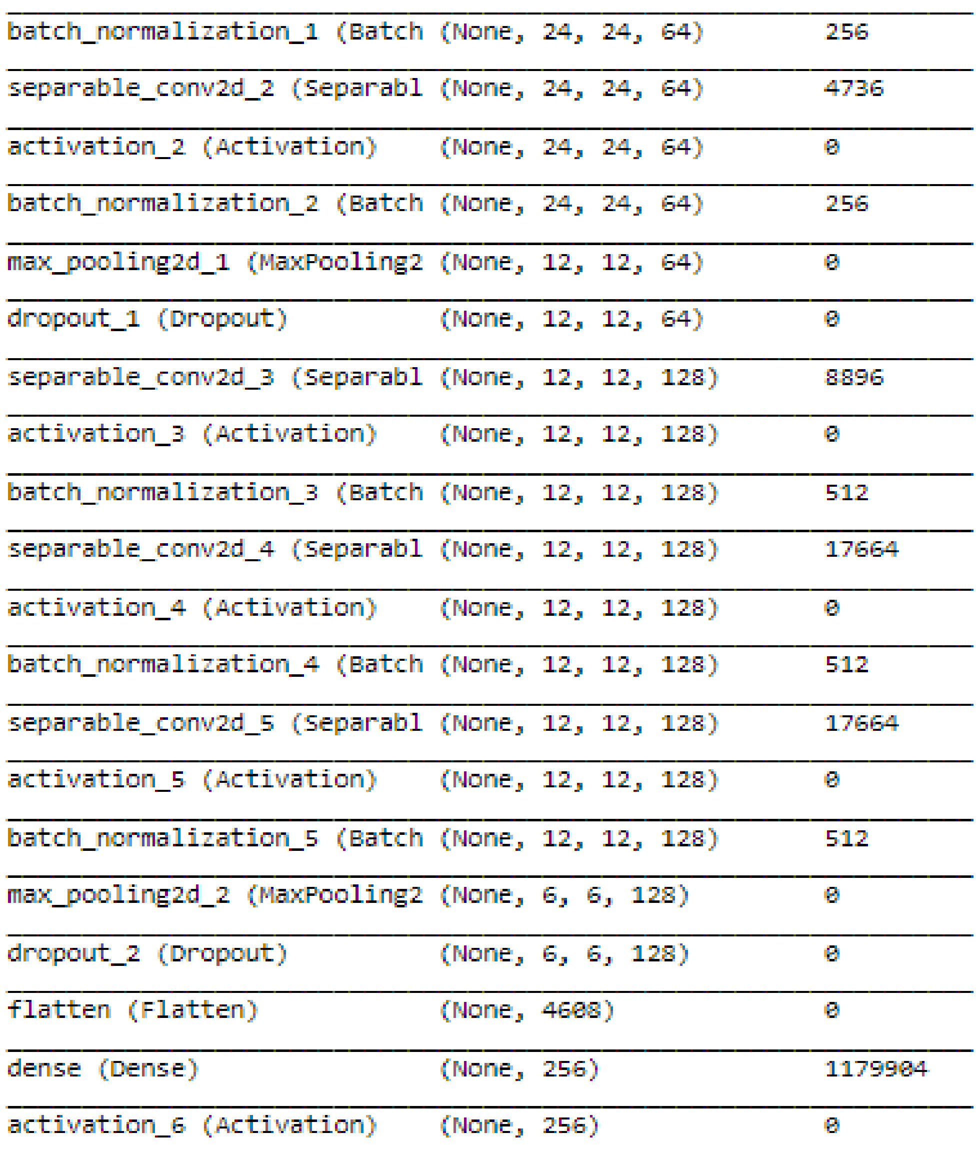
In[10]:

```
# Print the model summary
model.summary()
```

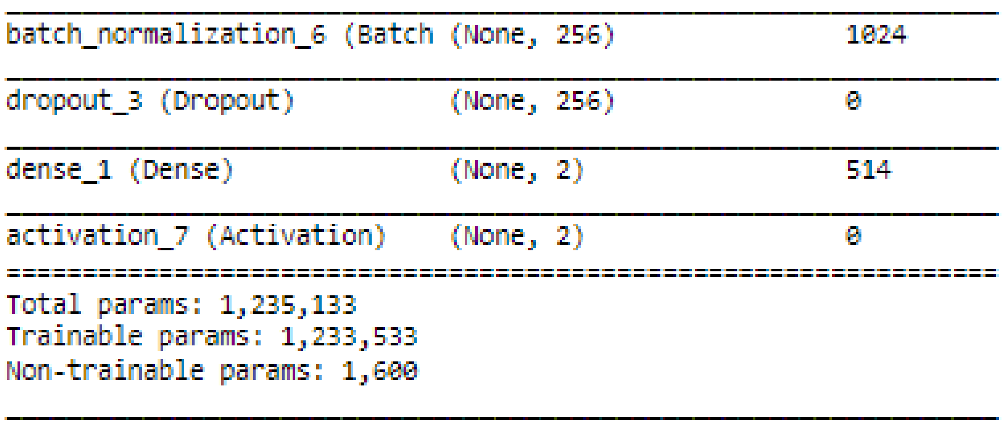
Out[10]:



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
separable_conv2d (SeparableC (None, 48, 48, 32)        155
_________________________________________________________________
activation (Activation)      (None, 48, 48, 32)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 48, 48, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 24, 24, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 24, 24, 32)        0
_________________________________________________________________
separable_conv2d_1 (Separabl (None, 24, 24, 64)        2400
_________________________________________________________________
activation_1 (Activation)    (None, 24, 24, 64)        0
```



```
_________________________________________________________________
batch_normalization_1 (Batch (None, 24, 24, 64)        256
_________________________________________________________________
separable_conv2d_2 (Separabl (None, 24, 24, 64)        4736
_________________________________________________________________
activation_2 (Activation)    (None, 24, 24, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 24, 24, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
separable_conv2d_3 (Separabl (None, 12, 12, 128)       8896
_________________________________________________________________
activation_3 (Activation)    (None, 12, 12, 128)       0
_________________________________________________________________
batch_normalization_3 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
separable_conv2d_4 (Separabl (None, 12, 12, 128)       17664
_________________________________________________________________
activation_4 (Activation)    (None, 12, 12, 128)       0
_________________________________________________________________
batch_normalization_4 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
separable_conv2d_5 (Separabl (None, 12, 12, 128)       17664
_________________________________________________________________
activation_5 (Activation)    (None, 12, 12, 128)       0
_________________________________________________________________
batch_normalization_5 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 256)               1179904
_________________________________________________________________
activation_6 (Activation)    (None, 256)               0
```



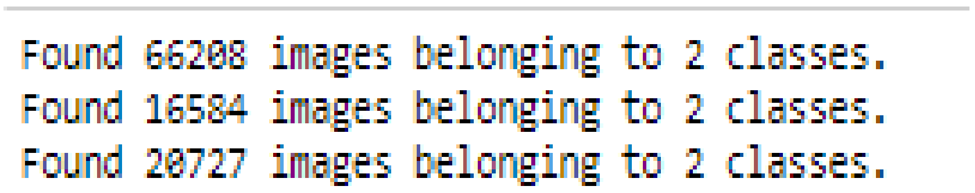
```
_________________________________________________________________
batch_normalization_6 (Batch (None, 256)               1024
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 514
_________________________________________________________________
activation_7 (Activation)    (None, 2)                 0
=================================================================
Total params: 1,235,133
Trainable params: 1,233,533
Non-trainable params: 1,600
_________________________________________________________________
```

In[11]:

```
# Get the data ready for modeling
import matplotlib
#matplotlib.use("Agg")
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import LearningRateScheduler
from keras.optimizer_v2 import adagrad
from keras.utils import np_utils
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os

NUM_EPOCHS=5
INIT_LR=1e-2
BS=32

trainPaths=list(paths.list_images(TRAIN_PATH))
lenTrain=len(trainPaths)
lenVal=len(list(paths.list_images(VAL_PATH)))
lenTest=len(list(paths.list_images(TEST_PATH)))

trainLabels=[int(p.split(os.path.sep)[-2]) for p in trainPaths]
trainLabels=np_utils.to_categorical(trainLabels)
```

```
classTotals=trainLabels.sum(axis=0)
classWeight=dict()
for i in range(0,len(classTotals)):
    classWeight[i] = classTotals.max()/classTotals[i]

trainAug = ImageDataGenerator(
  rescale=1/255.0,
  rotation_range=20,
  zoom_range=0.05,
  width_shift_range=0.1,
  height_shift_range=0.1,
  shear_range=0.05,
  horizontal_flip=True,
  vertical_flip=True,
  fill_mode="nearest")
valAug=ImageDataGenerator(rescale=1 / 255.0)
trainGen = trainAug.flow_from_directory(
  TRAIN_PATH,
  class_mode="categorical",
  target_size=(48,48),
  color_mode="rgb",
  shuffle=True,
  batch_size=BS)
valGen = valAug.flow_from_directory(
  VAL_PATH,
  class_mode="categorical",
  target_size=(48,48),
  color_mode="rgb",
  shuffle=False,
  batch_size=BS)
testGen = valAug.flow_from_directory(
  TEST_PATH,
  class_mode="categorical",
  target_size=(48,48),
  color_mode="rgb",
  shuffle=False,
  batch_size=BS)
```

Out[11]:

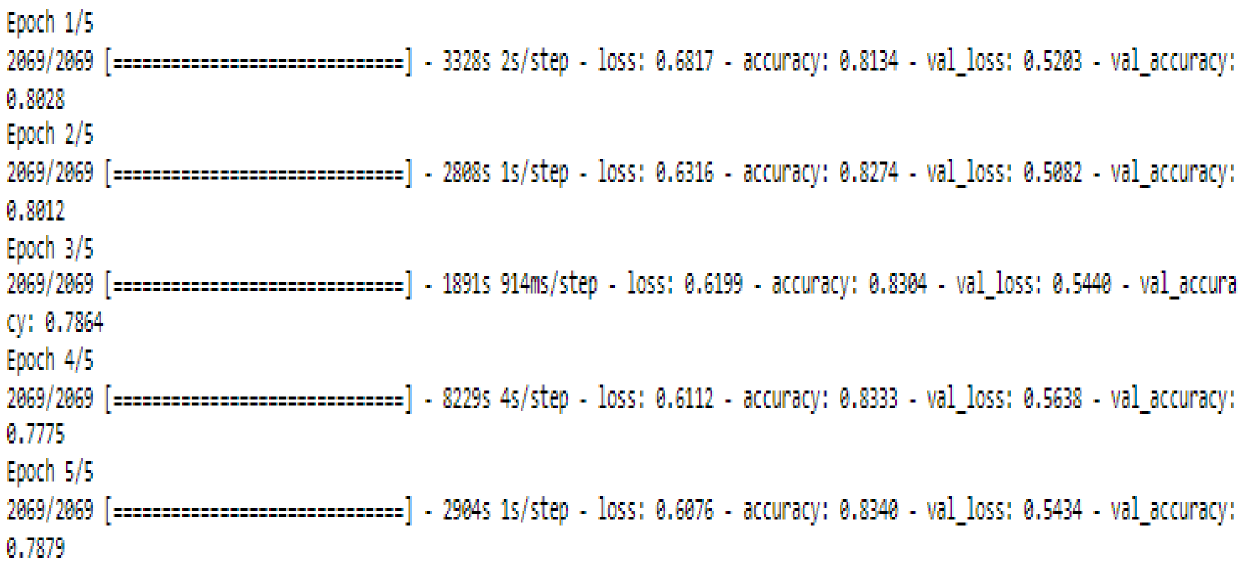


```
Found 66208 images belonging to 2 classes.
Found 16584 images belonging to 2 classes.
Found 20727 images belonging to 2 classes.
```

In[12]:

```
# Train the model
opt=keras.optimizer_v2.adagrad.Adagrad(lr=INIT_LR,decay=INIT
_LR/NUM_EPOCHS)
model.compile(loss="binary_crossentropy",optimizer=opt,metrics=
["accuracy"])

M=model.fit_generator(
  trainGen,
  steps_per_epoch=lenTrain//BS,
  validation_data=valGen,
  validation_steps=lenVal//BS,
  class_weight=classWeight,
  epochs=NUM_EPOCHS)
```

Out[12]:



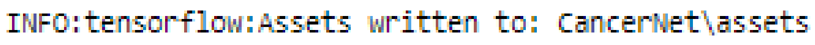
```
Epoch 1/5
2069/2069 [==============================] - 3328s 2s/step - loss: 0.6817 - accuracy: 0.8134 - val_loss: 0.5203 - val_accuracy:
0.8028
Epoch 2/5
2069/2069 [==============================] - 2808s 1s/step - loss: 0.6316 - accuracy: 0.8274 - val_loss: 0.5082 - val_accuracy:
0.8012
Epoch 3/5
2069/2069 [==============================] - 1891s 914ms/step - loss: 0.6199 - accuracy: 0.8304 - val_loss: 0.5440 - val_accura
cy: 0.7864
Epoch 4/5
2069/2069 [==============================] - 8229s 4s/step - loss: 0.6112 - accuracy: 0.8333 - val_loss: 0.5638 - val_accuracy:
0.7775
Epoch 5/5
2069/2069 [==============================] - 2904s 1s/step - loss: 0.6076 - accuracy: 0.8340 - val_loss: 0.5434 - val_accuracy:
0.7879
```

In[13]:

```
# Save the trained model
model.save('CancerNet')

# Use below to reload model
#model=keras.models.load_model("CancerNet")

import pickle
# Store off the history for later
```

```
with open('trainHistoryDict', 'wb') as file_pi:
        pickle.dump(M.history, file_pi)

# Use below to read
#history = pickle.load(open('/trainHistoryDict'), "rb")
```

Out[13]:



```
INFO:tensorflow:Assets written to: CancerNet\assets
```

In[14]:

```
# Additional training if needed
#model2=model
#NUM_EPOCHS=30
#M3=model2.fit_generator(
#  trainGen,
#  steps_per_epoch=lenTrain//BS,
#  validation_data=valGen,
#  validation_steps=lenVal//BS,
#  class_weight=classWeight,
#  epochs=NUM_EPOCHS)

#model2.save('./CancerNet21-50')
#with open('trainHistoryDict_21-50', 'wb') as file_pi:
#       pickle.dump(M3.history, file_pi)
```

In[15]:

```
# Combine histories if model was trained multiple times.
def appendHist(h1, h2):
    if h1 == {}:
        return h2
    else:
        dest = {}
        for key, value in h1.items():
            dest[key] = value + h2[key]
        return dest

#history = appendHist(M.history, M3.history)

history=M.history
print(len(history["accuracy"]))
```

Out[15]:

```
5
```

In[16]:

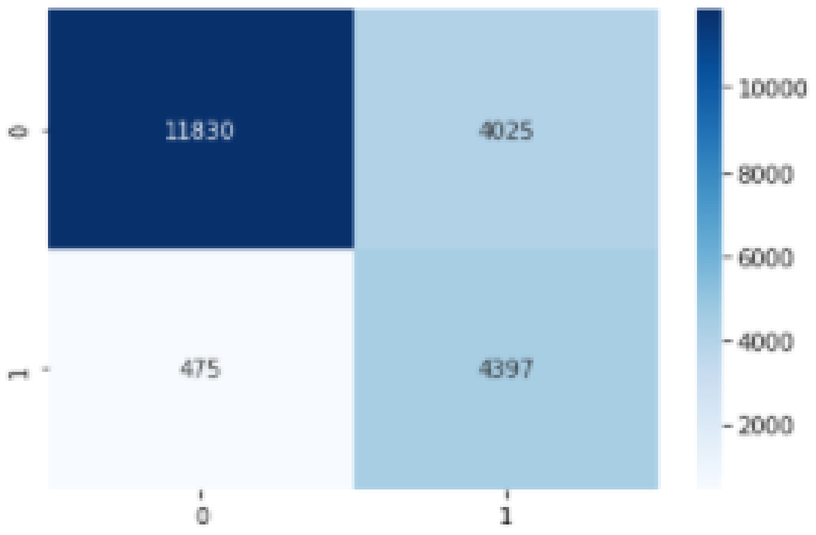
```
# Let's test the model and see how it performed
testGen.reset()
pred_indices=model.predict_generator(testGen,steps=(lenTest//BS)
+1)
pred_indices=np.argmax(pred_indices,axis=1)
```

In[17]:

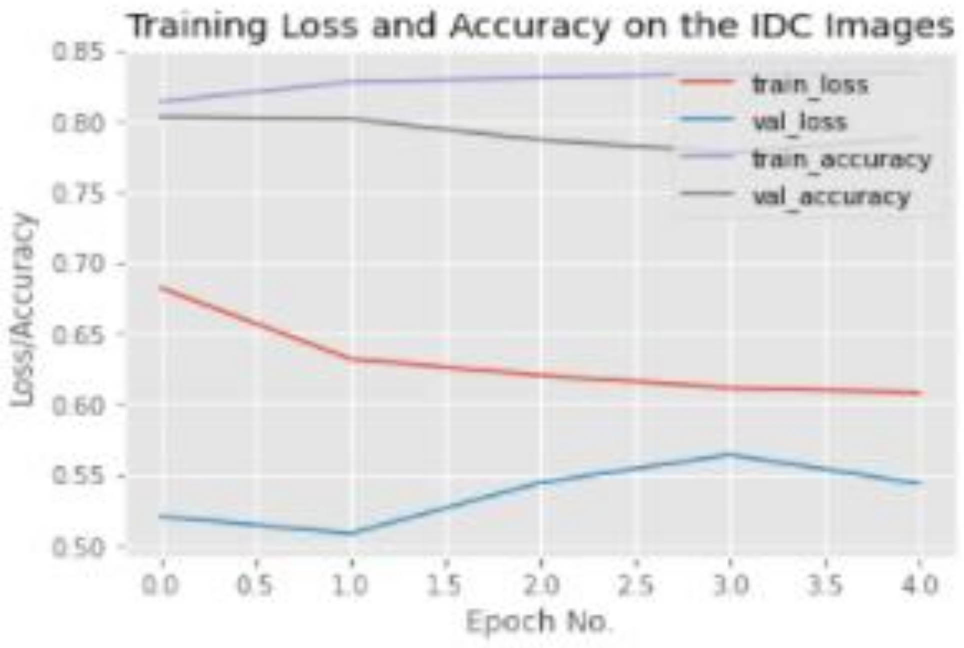
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, f1_score, recall_score
import seaborn as sns

print(classification_report(testGen.classes, pred_indices,
target_names=testGen.class_indices.keys()))
cm=confusion_matrix(testGen.classes,pred_indices)
plt.figure()
sns.heatmap(cm, annot=True,fmt='.0f', cmap="Blues")
plt.show()
#plt.savefig('3_Confusion_Matrix.png')
#score = model.evaluate(X_test, y_test)
#print(score)
print(accuracy_score(testGen.classes, pred_indices))
print(precision_score(testGen.classes, pred_indices))
print(recall_score(testGen.classes, pred_indices))
print(f1_score(testGen.classes, pred_indices))
total=sum(sum(cm))
accuracy=(cm[0,0]+cm[1,1])/total
specificity=cm[1,1]/(cm[1,0]+cm[1,1])
sensitivity=cm[0,0]/(cm[0,0]+cm[0,1])
print(cm)
print(f'Accuracy: {accuracy}')
print(f'Specificity: {specificity}')
print(f'Sensitivity: {sensitivity}')
#N = NUM_EPOCHS
N = len(history["accuracy"])
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0,N), history["loss"], label="train_loss")
plt.plot(np.arange(0,N), history["val_loss"], label="val_loss")
plt.plot(np.arange(0,N),                    history["accuracy"],
label="train_accuracy")
plt.plot(np.arange(0,N),                history["val_accuracy"],
label="val_accuracy")
plt.title("Training Loss and Accuracy on the IDC Images")
plt.xlabel("Epoch No.")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="upper right")
plt.show()
#plt.savefig('3_plot.png')
```

Out[17]:

```
              precision    recall  f1-score   support

           0       0.96      0.75      0.84     15855
           1       0.52      0.90      0.66      4872

    accuracy                           0.78     20727
   macro avg       0.74      0.82      0.75     20727
weighted avg       0.86      0.78      0.80     20727
```



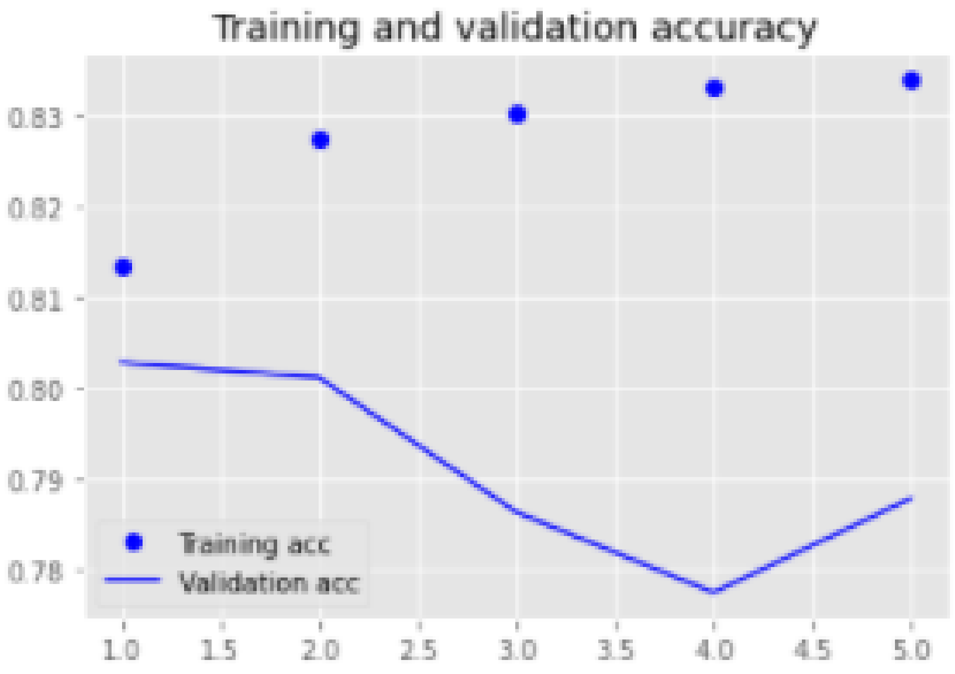
```
0.7828918801563178
0.5220850154357635
0.902504105090312
0.6615014292161878
[[11830  4025]
 [  475  4397]]
Accuracy: 0.7828918801563178
Specificity: 0.902504105090312
Sensitivity: 0.7461368653421634
```

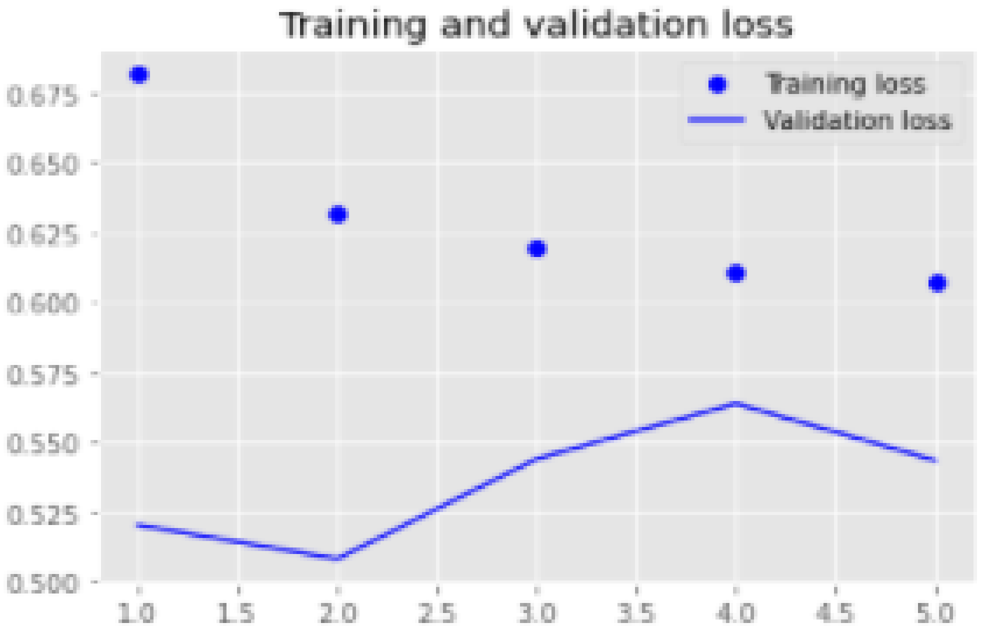


In[18]:

```
# Plot the model performance over time
acc = history['accuracy']
val_acc = history['val_accuracy']
loss = history['loss']
val_loss = history['val_loss']

epochs = range(1, len(acc) + 1)
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.show()
#plt.savefig('3_plot_acc.png')
```

```
plt.figure()

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.show()
#plt.savefig('3_plot_loss.png')

Out[18]:
```




Training and validation loss
Training loss
Validation loss
0.675
0.650
0.625
0.600
0.575
0.550
0.525
0.500
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

CHAPTER 8
CONCLUSION AND FUTURE SCOPE

In this paper , we have taken the histology images of the IDC patient tissues and used them to train Keras based Convolutional Neural Network to be able to predict if a particular histology images in benign or malignant. We have established that the CNN model has done a good job in predicting the classification of histology images with 78.28% accuracy. To understand even more, we have added specificity and sensitivity. Specificity is true negatives(90.25%) and sensitivity is true positives(74.61%). We believe that if we have used full dataset , we might get even more accuracy.

In future, as an improvement to the proposed method, one can implement an autoencoder instead of manually reducing image size. It can compress data without losing the prominent features, because autoencoders can regenerate up to 90% of the original image. From the point of method improvement, we can incorporate spectral imaging. Spectral imaging is used to obtain images with various wavelengths, which is different from the trivial three channel RGB image. Additionally, we may combine various imaging technologies such as MRI, CT Scan, ultrasound, and mammographic images, and determine their collective results. This technique is known as multimodel fusion. Problems stated above can again readily be solved by deep learning, and can be used to perform high quality research that might provide even better results.

Breast cancer causes great damage in women. Hence it is very important to prevent or take care, detect and get rid of the disease.

Automating the detection of breast cancer to enhance the care of patients is a challenging task. The current study proposes a CNN approach that analyses the IDC tissue regions in WSIs for the automatic detection of this cancer. Three different CNN architectures have been described in this paper with a proper comparison. The proposed system using CNN Model achieves 78% accuracy. Although Model 3 is deeper than Models 1 and 2, the five-layer CNN in Model 3 is best suited for this task. All architectures were guided by a big dataset of about 1,03,303 50×50-pixel RGB image patches. When we compared the proposed model with the machine learning (ML) algorithm. The proposed model was found to successfully obtain correct results that might decrease human mistakes in the diagnosis process and reduce the cost of cancer diagnosis. The main limitation of this study is to use the secondary database like Kaggle, and future study should be done based on primary data for more accuracy of the results related to breast cancer identification.

REFERENCES

REFERENCES

1] https://www.kaggle.com/paultimothymooney/predict-idc-in-breast-cancer-histology-images

2] https://www.kdnuggets.com/2019/10/convolutional-neural-network-breast-cancer-classification.html

3] https://www.hindawi.com/journals/jhe/2021/5528622/

4] https://www.analyticsvidhya.com/blog/2021/06/breast-cancer-classification-using-deep-learning/

5] https://data-flair.training/blogs/project-in-python-breast-cancer-classification/

6] https://www.slideshare.net/irjetjournal/breast-cancer-detection-using-convolution-neural-network

7] https://data-flair.training/blogs/convolutional-neural-networks-tutorial/

8] https://link.springer.com/chapter/10.1007/978-3-319-93000-8_83

9] https://ieeexplore.ieee.org/abstract/document/8308076/

10] https://ieeexplore.ieee.org/abstract/document/8457948/

11] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.017754