An Industrial Oriented Mini Project (CS704PC)

on

TO ANALYZE BREAST CANCER AND THEIR SEVERITY RANGE USING AN INTELLIGENT DEEP NEURAL MODEL

Submitted in partial fulfillment of the requirements for

the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Shabbir Jodhpurwala

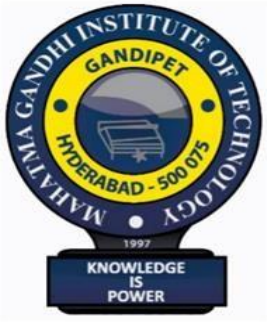
19261A05A7

Under the guidance

of

Mr. P Satya Shekar Varma

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)

Gandipet, Hyderabad-500 075, Telangana (India)

2022-2023

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)

GANDIPET, HYDERABAD – 500075, Telangana (INDIA)

CERTIFICATE

This is to certify that the project entitled "**TO ANALYZE BREAST CANCER STAGES AND THEIR SEVERITY RANGE USING AN INTELLIGENT DEEP NEURAL MODEL"** is being submitted by **Mr. Shabbir Jodhpurwala** bearing Roll No: **19261A05A7** in partial fulfilment of the requirements for the Industrial Oriented Mini Project (CS704PC) in **COMPUTER SCIENCE AND ENGINEERING** to **Jawaharlal Nehru Technological University Hyderabad** is a record of bonafide work carried out by him under our guidance and supervision.

The results of the investigations enclosed in this report have been verified and found satisfactory.

Project Guide	Head of the Department
Mr. P Satya Shekar Varma	**Dr. C.R.K. Reddy**
Assistant Professor	Professor

External Examiner

DECLARATION

This is to certify that the work reported in the project titled "**To Analyze breast cancer and their severity using an intelligent deep neural model** " is a record of work done by me in the Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the work done entirely by me and not copied from any other source.

SHABBIR JODHPURWALA

19261A05A7

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible because success is the abstract of hard work and perseverance, but steadfast of all is encouraging guidance. So, I acknowledge all those whose guidance and encouragement served as a beacon light and crowned my efforts with success.

I would like to express my sincere thanks to my guide **Mr. P. Satya Shekar Varma**, **Assistant Professor**, Department of CSE, MGIT, for his constant guidance, encouragement and moral support throughout the project.

I am extremely thankful to **Dr. C R K Reddy**, **Professor and HOD**, **Dr.B. Prasanthi**, **Assistant Professor**, Department of CSE, Industrial Oriented Mini Project Coordinators, MGIT, for their encouragement and support throughout the project.

I convey my heartfelt thanks to **Dr. C.R.K Reddy**, **Professor & HOD**, Department of Computer Science and Engineering, MGIT, for all the timely support and valuable suggestions during the period of project.

I am thankful to **Prof.G.ChandraMohan Reddy, Principal MGIT**, for providing the work facilities in the college.

Finally, I would like to thank all the faculty and staff of CSE Department who helped me directly or indirectly, for completing this project.

SHABBIR JODHPURWALA

19261A05A7

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Breast cancer is a leading cause of cancer-related deaths among women worldwide, and early detection of the disease is crucial for improving treatment outcomes. However, current methods for detecting breast cancer can be expensive and inconvenient, leading to low rates of screening among certain populations. In this project, we propose the development of a machine learning-based system for detecting the stages of breast cancer from medical images. Our system will be trained on a large dataset of breast cancer images and will utilize Convulted Neural Network using DenseNet and Relu as activation Function the model is compiled using the Adam optimizer, the Sparse Categorical Crossentropy loss function, and the accuracy metric to accurately classify the stage of breast cancer present in a given image. By providing a reliable and cost-effective means of detecting breast cancer stages, we hope to improve access to early detection and treatment of the disease, ultimately reducing the incidence and mortality rates of breast cancer.

Keywords: Sequential Model,DenseNet Model,Convolutional Neural networks.

1. INTRODUCTION

Breast cancer is one of the major causes of death in women around the world. According to the American cancer society, 41,760 women and more than 500 men died from breast cancer recently. There are three main parts in which the breast is made up of lobules, ducts and connective tissue. The inside of the breast is divided into about twenty different sections called lobes. Each lobe is further divided into lobules. The lobules are the glands that make up the milk. Milk-producing glands place the milk in tiny ducts. Milk ducts arerun the length of the breast and store the milk in a chamber underneath the nipple. The connective tissue encircles and holds everything together. The connective tissue consists of fat and fibrous tissues.When healthy breast cells get out of balance and form a tumour, which is a mass or layer of cells, breast cancer is diagnosed. Cancerous and noncancerous tumours exist. Breast Cancer stages are classifies using TNM staging system:

1. T categories for breast cancer

T followed by a number from 0 to 4 describes the main (primary) tumor's size and if it has spread to the skin or to the chest wall under the breast. Higher T numbers mean a larger tumor and/or wider spread to tissues near the breast.

TX: Primary tumor cannot be assessed.

T0: No evidence of primary tumor.

Tis: Carcinoma in situ (DCIS, or Paget disease of the breast with no associated tumor mass)

T1 (includes T1a, T1b, and T1c): Tumor is 2 cm (3/4 of an inch) or less across.

T2: Tumor is more than 2 cm but not more than 5 cm (2 inches) across.

T3: Tumor is more than 5 cm across.

T4 (includes T4a, T4b, T4c, and T4d): Tumor of any size growing into the chest wall or skin. This includes inflammatory breast cancer.

2. N categories for breast cancer

N followed by a number from 0 to 3 indicates whether the cancer has spread to lymph nodes near the breast and, if so, how many lymph nodes are involved.

Lymph node staging for breast cancer is based on how the nodes look under the microscope, and has changed as technology has gotten better. Newer methods have made it possible to find smaller and smaller groups of cancer cells, but experts haven't been sure how much these tiny deposits of cancer cells influence outlook.

It’s not yet clear how much cancer in the lymph node is needed to see a change in outlook or treatment. This is still being studied, but for now, a deposit of cancer cells must contain at least

200 cells or be at least 0.2 mm across (less than 1/100 of an inch) for it to change the N stage. An area of cancer spread that is smaller than 0.2 mm (or fewer than 200 cells) doesn't change the stage, but is recorded with abbreviations (i+ or mol+) that indicate the type of special test used to find the spread.

If the area of cancer spread is at least 0.2 mm (or 200 cells), but still not larger than 2 mm, it is called a micrometastasis (one mm is about the size of the width of a grain of rice). Micrometastases are counted only if there aren't any larger areas of cancer spread. Areas of cancer spread larger than 2 mm are known to influence outlook and do change the N stage. These larger areas are sometimes called macrometastases, but are more often just called metastases.

NX: Nearby lymph nodes cannot be assessed (for example, if they were removed previously).

N0: Cancer has not spread to nearby lymph nodes.

N0(i+): The area of cancer spread contains fewer than 200 cells and is smaller than 0.2 mm. The abbreviation "i+" means that a small number of cancer cells (called isolated tumor cells) were seen in routine stains or when a special type of staining technique, called *immunohistochemistry*, was used.

N0(mol+): Cancer cells cannot be seen in underarm lymph nodes (even using special stains), but traces of cancer cells were detected using a technique called *RT-PCR*. RT-PCR is a molecular test that can find very small numbers of cancer cells.

N1: Cancer has spread to 1 to 3 axillary (underarm) lymph node(s), and/or cancer is found in internal mammary lymph nodes (those near the breast bone) on sentinel lymph node biopsy.

N1mi: Micrometastases (tiny areas of cancer spread) in the lymph nodes under the arm. The areas of cancer spread in the lymph nodes are at least 0.2mm across, but not larger than 2mm.

N1a: Cancer has spread to 1 to 3 lymph nodes under the arm with at least one area of cancer spread greater than 2 mm across.

N1b: Cancer has spread to internal mammary lymph nodes on the same side as the cancer, but this spread could only be found on sentinel lymph node biopsy (it did not cause the lymph nodes to become enlarged).

N1c: Both N1a and N1b apply.

N2: Cancer has spread to 4 to 9 lymph nodes under the arm, or cancer has enlarged the internal mammary lymph nodes

N2a: Cancer has spread to 4 to 9 lymph nodes under the arm, with at least one area of cancer spread larger than 2 mm.

N2b: Cancer has spread to one or more internal mammary lymph nodes, causing them to become enlarged.

N3: Any of the following:

N3a: either:

Cancer has spread to 10 or more axillary lymph nodes, with at least one area of cancer spread greater than 2 mm,

OR

Cancer has spread to the lymph nodes under the collarbone (infraclavicular nodes), with at least one area of cancer spread greater than 2 mm.

N3b: either:

Cancer is found in at least one axillary lymph node (with at least one area of cancer spread greater than 2 mm) and has enlarged the internal mammary lymph nodes,

OR

Cancer has spread to 4 or more axillary lymph nodes (with at least one area of cancer spread greater than 2 mm), and to the internal mammary lymph nodes on sentinel lymph node biopsy.

N3c: Cancer has spread to the lymph nodes above the collarbone (supraclavicular nodes) on the same side of the cancer with at least one area of cancer spread greater than 2 mm.

3. M categories for breast cancer

M followed by a 0 or 1 indicates whether the cancer has spread to distant organs -- for example, the lungs, liver, or bones.

M0: No distant spread is found on x-rays (or other imaging tests) or by physical exam.

cM0(i+): Small numbers of cancer cells are found in blood or bone marrow (found only by special tests), or tiny areas of cancer spread (no larger than 0.2 mm) are found in lymph nodes away from the underarm, collarbone, or internal mammary areas.

M1: Cancer has spread to distant organs (most often to the bones, lungs, brain, or liver) as seen on imaging tests or by physical exam, and/or a biopsy of one of these areas proves cancer has spread and is larger than 0.2mm.

Breast cancer may be categorized as either invasive (spreading) or non-invasive (not spreading).

1.1 Problem Definition

In order to reduce breast cancers ,we are diagnosing and classifying the Breast Cancer Disease using Deep Neural Networks, Training and Testing the Dataset images using models like DenseNet Model and Sequential Model.

1.2 Existing System

The existing model consists of

ResNet model ,SVM(support vector Machine),KNN ,Decision Tree, ANN,BPNN,RBF etc.

Feature Selected:

- Size
- Frequency
- Shapes

Limitations:

- Less Accuracy
- Requires more time to train
- requires heavy GPU computational power

1.3 Proposed System

Proposed models is

DNN models like DenseNet,Sequential Model by using Neural Networks and Adam Optimizer.

Feature Selected:

- Size
- Frequency
- Shape

Proposed Improvements:

- Better Accuracy
- less computational power
- reduced training time

1.4 Requirements Specification

1.4.1 Software Requirements

- **Operating System**:Windows/Linux
- **Tools:**Google Colab

1.4.2 Hardware Requirements

- **RAM:**4 GB
- **Memory:**64 GB
- **Input Devices:** Keyboard, Mouse

2. LITERATURE SURVEY

[1] K.Kousalya and T.Saranya,“ Improved detection and classification of breast cancer using hyper parameter tuning” in 2021.

A normal machine learning algorithm was used, svm was used but the accuracy plummeted rapidly when more complex data sets were used the model tend to overtrain and learn the dataset.

[2] Dayong Wang ,Aditya Khosla ,Rishab Gargeya, Humayun Irshad, Andrew H Beck, “Deep Learning for Identifying Metastatic Breast Cancer” in 2019

In this paper propose a patch-based classification stage and a heatmap-based postprocessing stage. The patch based classification stage uses as input 256x256 pixel patches from positive and negative regions of the WSIs and trains linear regression classification model to discriminate between the positive and negative patches.

[3] Xin Feng and Lelian Song ,“ Accurate Prediction of Neoadjuvant Chemotherapy Pathological Complete Remission (pCR) for the Four Sub-Types of Breast Cancer” in 2019.

A machine Learning algorithm i.e KNN ,SVM,N Bayes,DTree,RF and XGB is used to predict breast cancer.All this method are used and the results are compared each individual model yields variable results , the algorithm KNN achieved the best Avc = 0.5989 and after some improvement XGB improved Avc to 0.7183.

[4] Adam Yala , Constance Lehman, “A Deep Learning Mammography-based Model for Improved Breast Cancer Risk Prediction” in 2019.

In this paper a risk-factor-based logistic regression model (RF-LR) that used traditional risk factors, a DL model (image-only DL) that used mammograms alone, and a hybrid DL model that used both traditional risk factors and mammograms. Comparisons were made to an established breast cancer risk model that included breast density. Model performance was compared by using areas under the receiver operating characteristic curve (AUCs) with DeLong test ($P < .05$). Hybrid DL and image-only DL

showed AUCs of 0.70 (95% confidence interval [CI]: 0.66, 0.75) and 0.68 (95% CI: 0.64, 0.73), respectively. RF-LR and TC showed AUCs of 0.67 (95% CI: 0.62, 0.72) and 0.62 (95% CI: 0.57, 0.66), respectively. Hybrid DL showed a significantly higher AUC (0.70) than TC (0.62; P < .001) and RF-LR (0.67; P = .01).

[5] Byungjae Lee and Kyunghyun Paeng, "A Robust and Effective Approach Towards Accurate Metastasis Detection and pN-stage Classification in Breast Cancer" in 2018.

In this paper they have try to predict pN-stage from lymph node histological slides, using CNN based metastasis detection and random forest based lymph node classification using Camelyon17 dataset. It requires more time for training and validation of CNN algorithms.

Table 2.1: The below table shows the Literature Survey of Diagnosis and classification of breast cancer disease.

S. No	Title of the Referred paper	Name of the Authors	year	Algorithm/Technique/Methodology used in the paper	Demerits
1	Improved detection and classification of breast cancer using hyper parameter tuning	K.Kousalya and T.Saranya	2021	SVM and Artificial Intelligence	The model tend to overtrain and did not yield good results.
2	Deep Learning for Identifying Metastatic Breast Cancer	Dayong Wang ,Aditya Khosla ,Rishab Gargeya, Humayun Irshad, Andrew H Beck	2019	Patch based classficiation using linear regression	Vanishing gradient problem.
3	Accurate Prediction of Neoadjuvant Chemotherapy Pathological Complete Remission (pCR) for the Four Sub-Types of Breast Cancer	Xin Feng and Lelian Song	2019	KNN, XGB	It occurs vanishing Gradient problem. Does not work well with larger dataset.

4	A Deep Learning Mammography-based Model for Improved Breast Cancer Risk Prediction	Adam Yala , Constance Lehman	2019	Logistic regression model	The performance of our system is lower for the extended dataset due to its increased complexity.
5.	A Robust and Effective Approach Towards Accurate Metastasis Detection and pN-stage Classification in Breast Cancer	Byungjae Lee,Kyunghyun Paeng	2018	CNN, and random forest	Requires more training time for training and Validation of CNN algorithms

3. DESIGN METHODOLOGY

3.1 Block diagram:

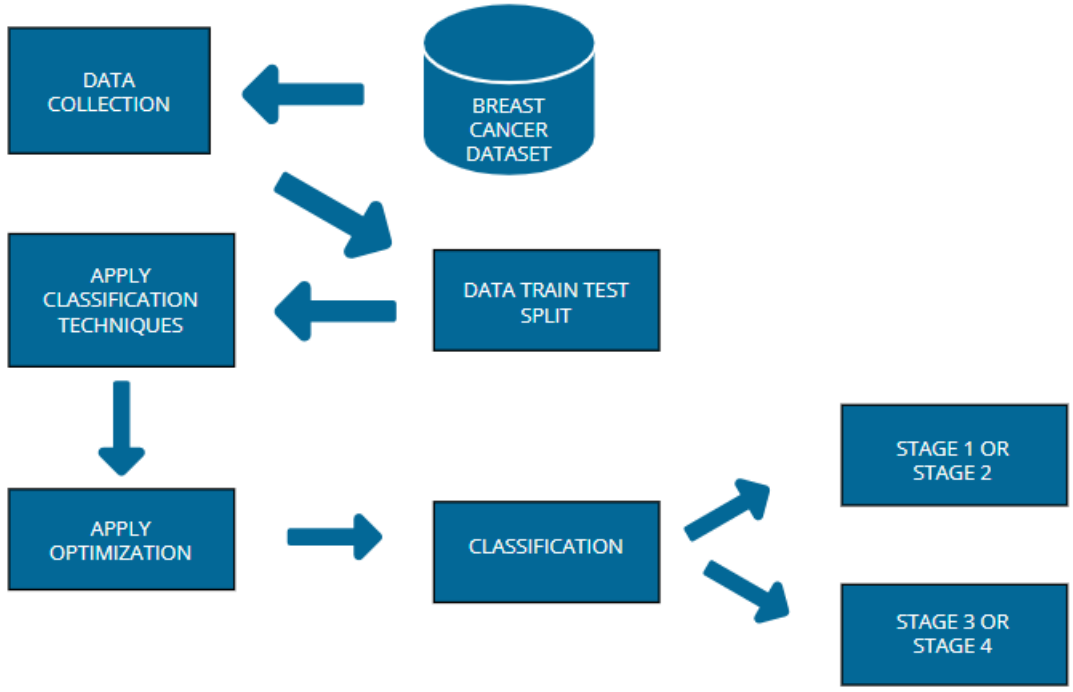


Figure 3.1: The above figure is the block diagram of diagnosis and classification of Breast cancer Disease.

The above block diagram fig 3.1 shows the block diagram of diagnosis and classification of breast cancer disease.

From the above figure ,first we have to take dataset and collect the images. Different image sizes are not accepted for Model Training, So, all images need to be transferred to some fixed size it is done by Image Preprocessing, in Image Preprocessing we have reshaping the image, crop, rotation, changing into RGB etc., can be done.

In the next step we divide the image data set into training and testing data sets. Next, using a preprocessed training image data set we train the model. After training the model we evaluate the model using testing image data set. After the evaluation of model we get accuracy which indicates how well the model can predict. The images given to the training dataset are used for training the model and when an input image is given, it is initially scaled to predefined size for easy computations and then Sequential model is used to classify the images and along with ResNet50 model.

In this project all the images are resized to the dimensions 224 x 224. After scaling, we get the images of the same size which helps in analyzing and testing the system easily. Then the data set is divided into training and testing sets.

The images are converted into numpy arrays in which all of its fields consist of the pixel values. All these arrays are grouped together as a Numpy array dataset and then they are labelled across each image indicating whether the image is affected by disease or not.

3.2 DataSet

Choosing the right dataset which would work best for the Sequential model required a little effort. The dataset used in training the model in a given approach was a combination of various open-source datasets and pictures, which included data from the Kaggle's Breast Images Dataset.

Kaggle dataset :Breast Histopathology Images For Stages

The Kaggle's Breast Cancer dataset which consists of 279 images from which 277,524 patches of size 50 x 50 were extracted and the test images are of 84 and the train images are of 195 images.

Figure 3.2: Above figures shows malignant and benign images for Breast Cancer

3.3 Testing

Testing is a very important module in software development to verify, validate and provide quality and service for different components of the software. It is used to minimize the risks by efficient use of resources in the development life cycle. It is efficient for the testing phase to be implemented at the initial level to lower down the risks. Software is tested and implemented in various conditions and environments to

examine a different aspect of the software. There exist various organizations for testing which is based on the type of software developed. This testing phase helps each and every module to work effectively and generate optimum results. In order to test data first, we have to train the data set using an algorithm. After training the dataset with algorithms we have to test the dataset with the same algorithms and check the accuracy of each algorithm and predicted values of each algorithm.

The system detects whether the given image is effected by disease or not. Testing is done by initially pre-processing the given input images by resizing them to desired size. Once the testing is done, the results of the test are given so that necessary actions can be taken.

Choosing the right dataset which would work best for the Sequential model and DenseNet Model required a little effort. The dataset used in training the model in a given approach was a combination of various open-source datasets and pictures, which included data from the Kaggle's Dataset.

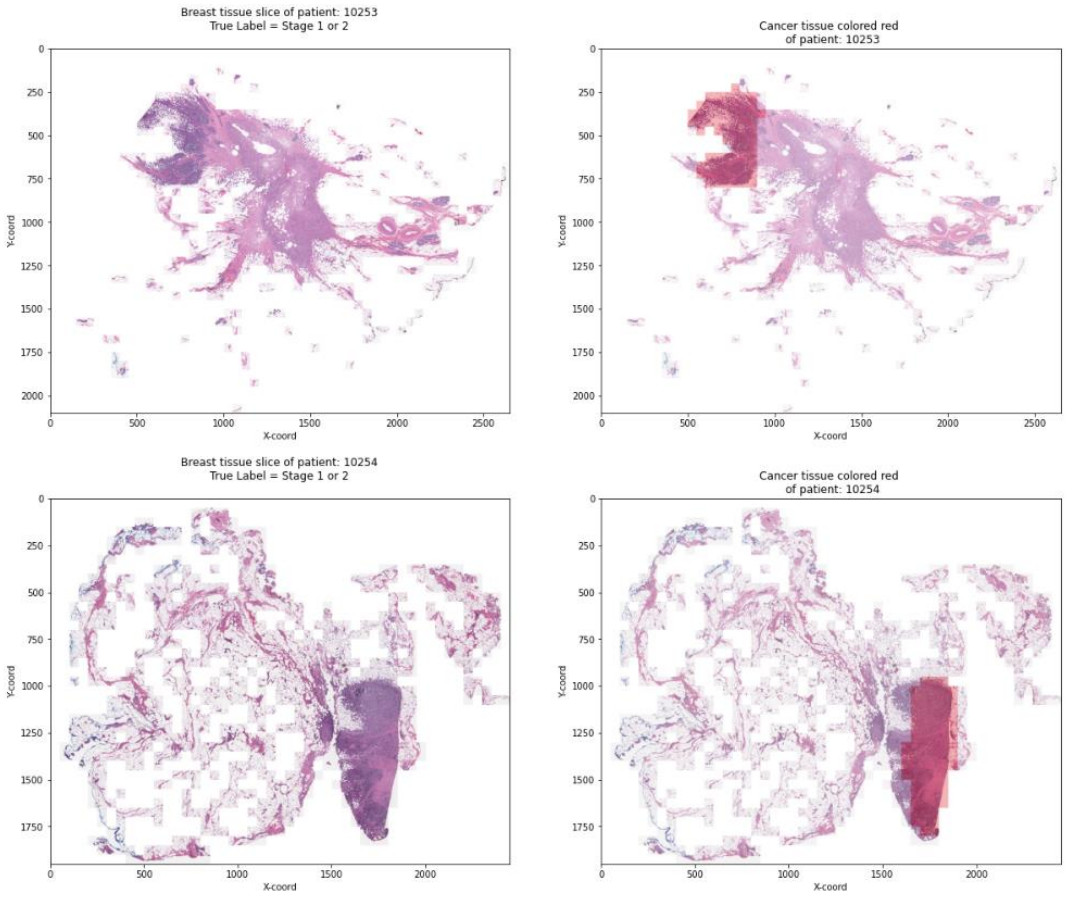


Figure 3.3: The above figure Shows test images of Breast cancer Disease

3.4 Training Report

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
```

```
Epoch 1/10
32/32 [==============================] - 3s 56ms/step - loss: 0.3400 - accuracy: 0.8730 - val_loss: 0.2740 - val_accuracy: 0.899
0
Epoch 2/10
32/32 [==============================] - 2s 48ms/step - loss: 0.2597 - accuracy: 0.9070 - val_loss: 0.3150 - val_accuracy: 0.882
0
Epoch 3/10
32/32 [==============================] - 2s 70ms/step - loss: 0.1869 - accuracy: 0.9310 - val_loss: 0.3360 - val_accuracy: 0.840
0
Epoch 4/10
32/32 [==============================] - 2s 50ms/step - loss: 0.1775 - accuracy: 0.9340 - val_loss: 0.3515 - val_accuracy: 0.874
0
Epoch 5/10
32/32 [==============================] - 1s 47ms/step - loss: 0.1648 - accuracy: 0.9290 - val_loss: 0.4458 - val_accuracy: 0.874
0
Epoch 6/10
32/32 [==============================] - 1s 46ms/step - loss: 0.1865 - accuracy: 0.9260 - val_loss: 0.3825 - val_accuracy: 0.827
0
Epoch 7/10
32/32 [==============================] - 2s 48ms/step - loss: 0.1631 - accuracy: 0.9390 - val_loss: 0.3599 - val_accuracy: 0.876
0
Epoch 8/10
32/32 [==============================] - 1s 47ms/step - loss: 0.1701 - accuracy: 0.9300 - val_loss: 0.4167 - val_accuracy: 0.860
0
Epoch 9/10
32/32 [==============================] - 2s 47ms/step - loss: 0.1697 - accuracy: 0.9380 - val_loss: 0.4015 - val_accuracy: 0.846
0
Epoch 10/10
32/32 [==============================] - 1s 47ms/step - loss: 0.1407 - accuracy: 0.9460 - val_loss: 0.4308 - val_accuracy: 0.824
0
```

Figure 3.4: The above figure shows the training report of Classifying the Breast Cancer Disease**s.**

In the above snippet I have taken 10 epochs and we get loss and accuracy of every epoch.

Training data is an extremely large dataset that is used to teach a deep learning model. Training data is used to teach prediction Model that use machine learning algorithms how to extract features that are relevant to specific business goals. For Supervised learning models, the training data is labeled. The data used to train unsupervised learning models is not labeled.

3.5 Importing Libraries

3.5.1 Matplotlib:

Matplotlib is a Python 2D plotting library which is used to plot a given image in a number ofsmall images and it will check with the dataset and it will produce publication quality figuresin a variety of hard copy formats and interactive environments across platforms.

3.5.2 TensorFlow:

PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab (FAIR). It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface.

3.5.3 Numpy:

NumPy is the fundamental package for scientific computing with Python. As it is used to divide the given image from a construction site into n dimensional objects such that it can be easily compared with the dataset.

3.5.4 DenseNet:

DenseNet is a type of convolutional neural network (CNN) that was developed to improve upon the performance of traditional CNNs on image classification tasks. One of the main features of DenseNet is that it introduces the concept of "dense blocks," which are groups of convolutional layers where each layer is connected to every other layer in the block. This dense connectivity allows the network to learn more robust and complex features by reusing features from earlier layers. Typical DenseNet models are implemented with double- or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between. DenseNets also use skip connections, which skip over one or more layers and directly combine the activations from earlier layers with those from later layers. This helps to alleviate the vanishing gradient problem, which is a common issue in deep neural networks where the gradients of the parameters with respect to the loss function become very small, making it difficult to update the parameters.

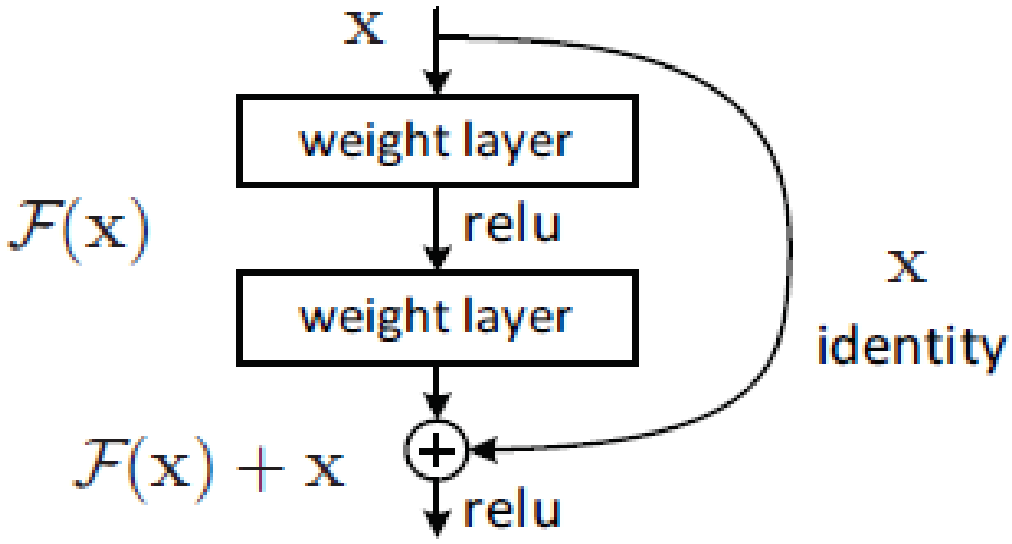


Figure 3.5: Skip connections In DenseNet Model

3.6 Deep Learning Techniques:

3.6.1 Convolutional Neural Networks

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

3.6.2 Sequential Model

Keras has come up with different types of in-built models. The Sequential model tends to be one of the simplest models as it constitutes a linear set of layers. The layers within the sequential models are sequentially arranged, so it is known as Sequential model. In most of the Artificial Neural Network, the layers are sequentially arranged, such that the data flow in between layers is in a specified sequence until it hit the output layer. Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. We use the 'add()' function to add layers to our model.

We can add Conv2D layers. These are convolution layers that will deal with our input images, which are seen as 2-dimensional matrices.32 in the first layer and 64 in the second layer are the number of nodes in each layer. This number can be adjusted to be higher or lower, depending on the size of the dataset.

Kernel size is the size of the filter matrix for our convolution. So a kernel size of 3 means we will have a 3x3 filter matrix. Activation is the activation function for the layer. This activation function has been proven to work well in neural networks. In between the Conv2D layers and the dense layer, there is a 'Flatten' layer. Flatten serves as a connection between the convolution and dense layers. 'Dense' is the layer type we will use in our output layer. Dense is a standard layer type that is used in many cases for neural networks.

We also have max Pooling layer in between each Convo2D layer which helps reduce the spatial size of the convolved feature and also helps reduce over-fitting by providing an abstracted representation of them.

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(50, 50, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(2))

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Figure 3.6 : The above figure shows the sequential model of Classification of Breast Cancer

3.7 UML Diagrams

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system. UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into

the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficiency.

3.7.1 Use Case Diagram

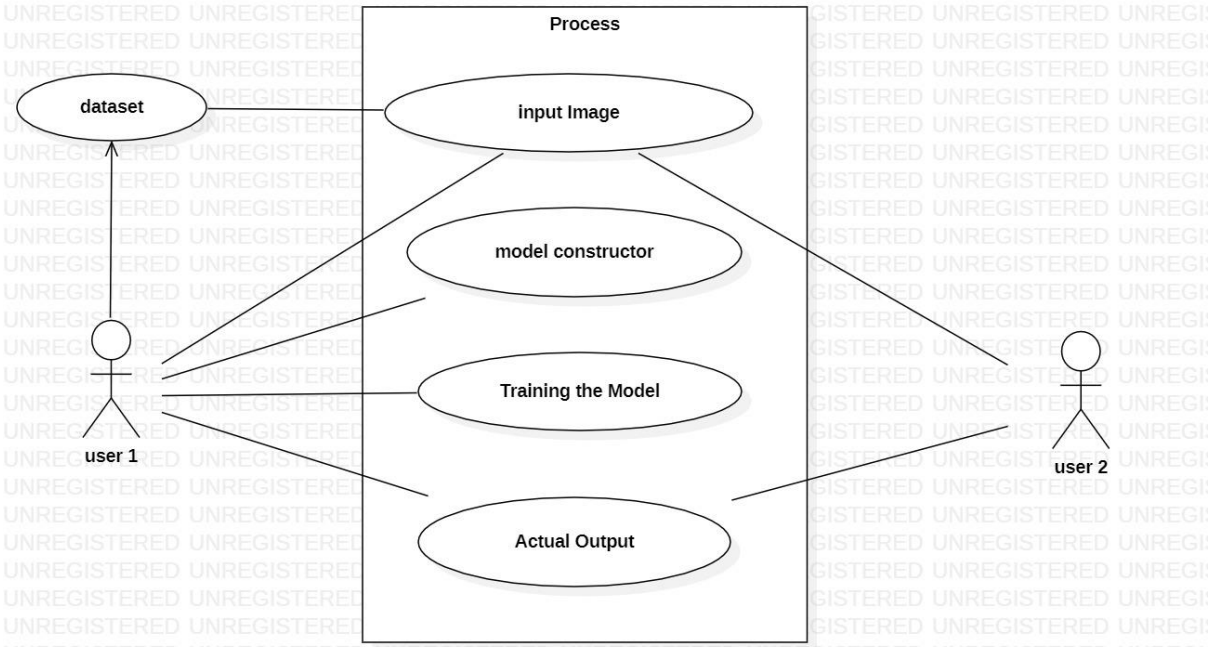


Figure 3.7: The above figure shows the use case diagram of the classification of Breast cancer diseases.

The above figure 3.7 is a use case diagram which is helpful in exposing the requirements and planning the project during the initial stage. The following figure describes the use case diagram for classifying Breast cancer Disease.

3.7.2 Activity Diagram

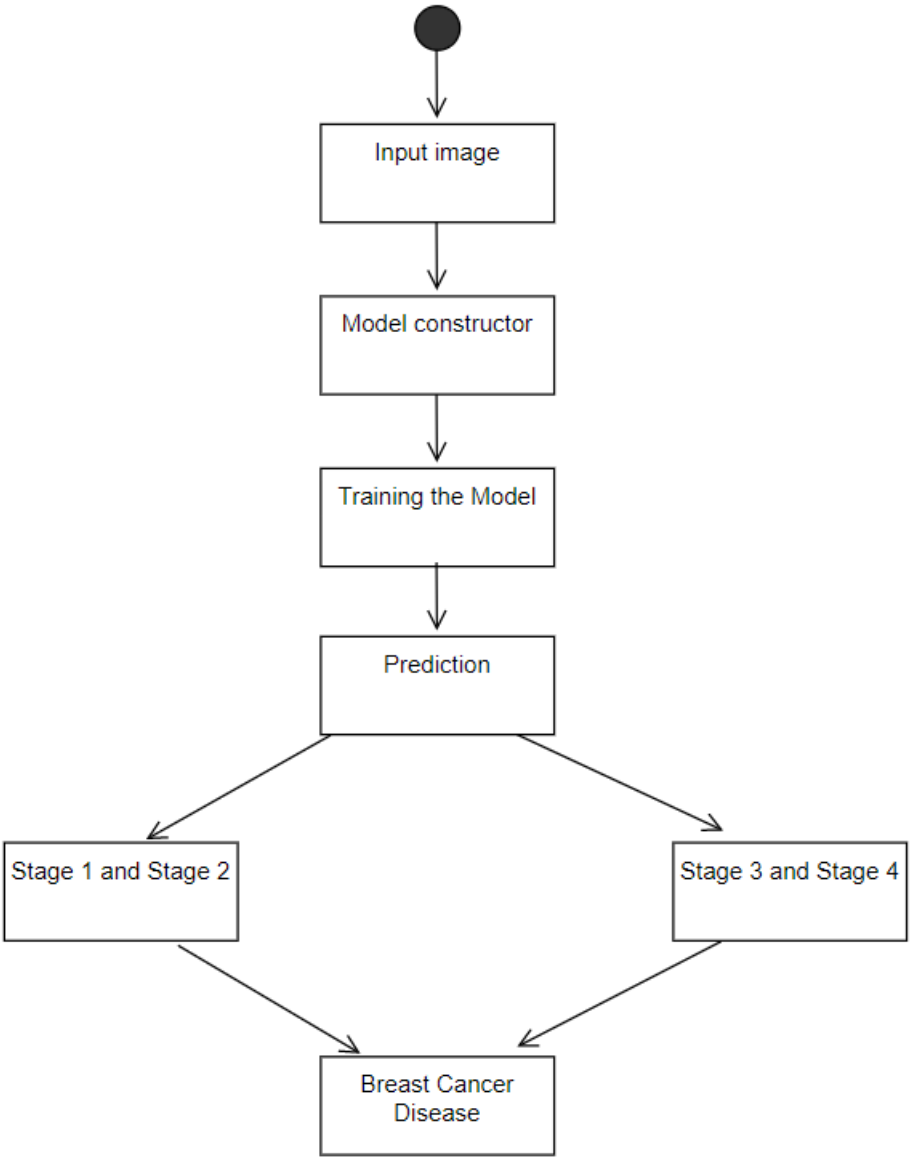


Figure 3.8: The above figure shows the activity diagram of Classification of Breast Cancer Diseases.

4. IMPLEMENTATION AND RESULTS

4.1 Implementation

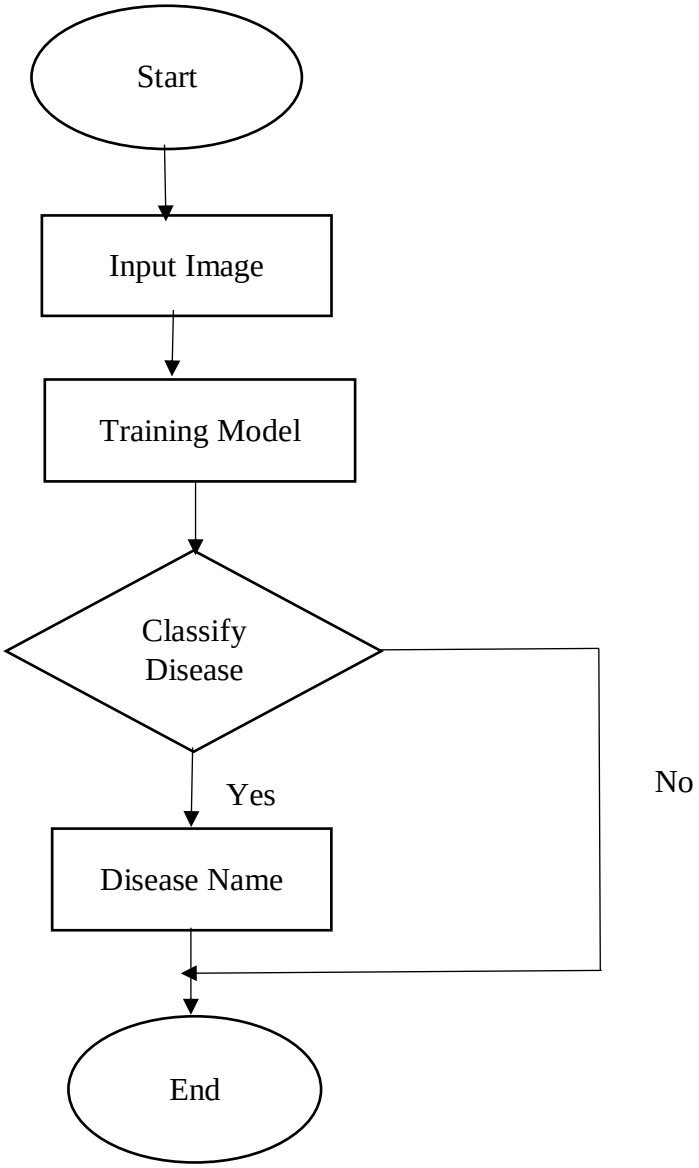


Figure 4.1: The above figure shows the flowchart of Classification of Breast Cancer

The Implementation of Classification of Breast Cancer Diseases involves the execution of the code in Kaggle Notebook.

1. Here we need to link the Dataset present in Kaggle.
2. In first step the preprocessing of image and to resized to the dimensions 224*224 takes place.
3. The second step is to create the model to the taken dataset and the sequential model and DenseNet Model is the output view of the model.

4. The other step is Training the model.

5. Final Result is the actual result of Stages of Cancer comparing with Predicted images.

4.2 Results:

The below figure 4.2 shows the predictions on some images. These are the classifications made by the Sequential model. Each and every image is classified to whether Stage1 and 2,or Stage 3 and 4.The accuracy of the Model is 92%,the model learns from the pattern of train dataset and labels and then makes classifications. These images are randomly chosen from a set of testing images folder .

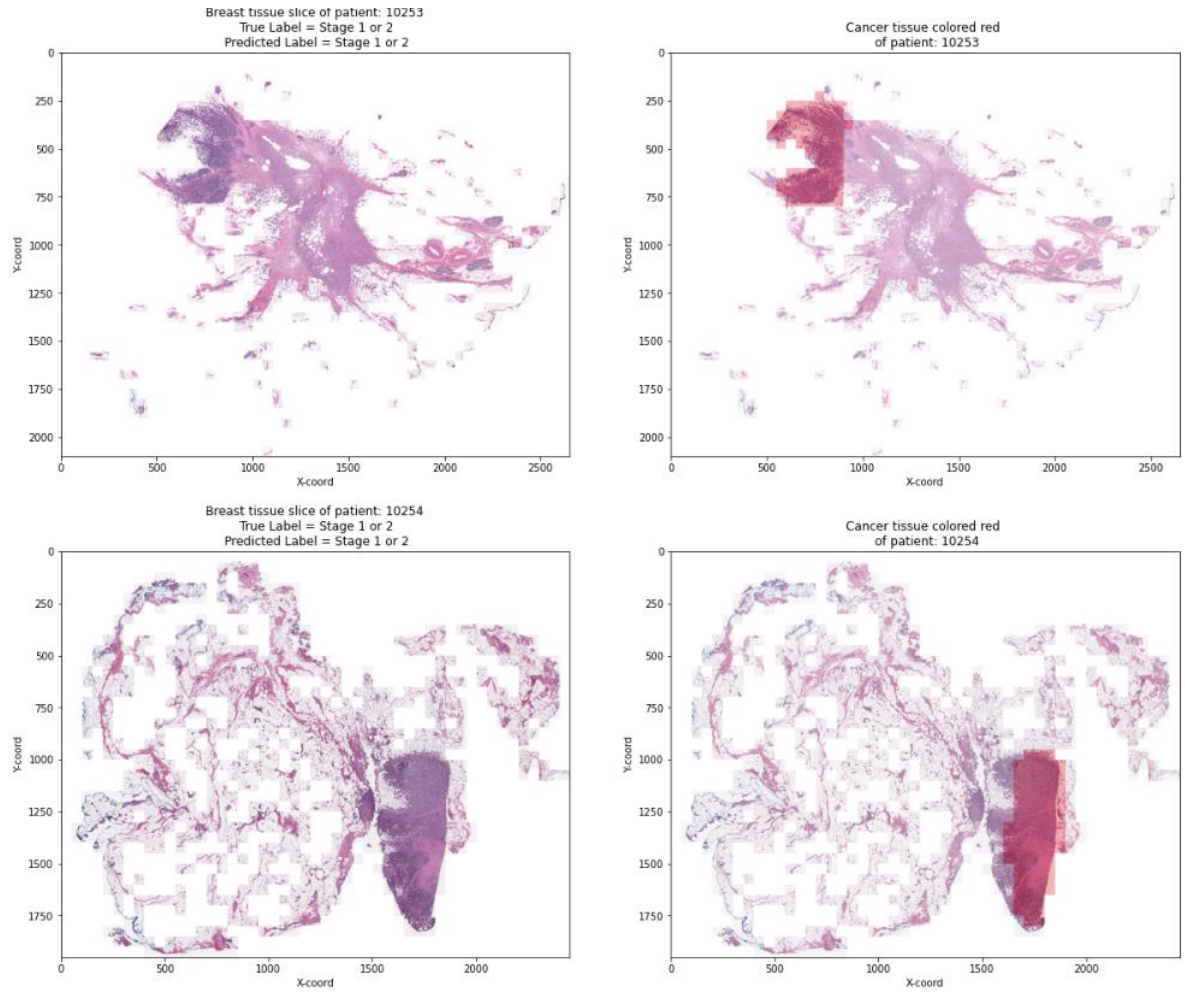


Figure 4.2: The above figure shows the Result of Classification of Breast Cancer Stages

5. CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

Breast cancer is one of the leading causes of death in women. Women's mortality rates can be lowered if breast cancer is diagnosed early and treated accordingly. As a consequence, we proposed this method for detecting and analysing the stages of breast cancer. In the proposed method, features that have been eligible, tested, and validated are extracted from breast histology photographs. The proposed approach derives features from breast histology photos that have been determined to be eligible, checked, and validated. In addition, 70% of images were used for training purposes and the remaining 30% were used for testing purposes. Performance evaluation was based on different matrices such as accuracy, precision, sensitivity, and specificity.The accuracy of this model using sequential model and DenseNet model is 92%.

5.2 Future scope

In Future we can use different models and algorithms which can increase performance of the model.In future we can handle larger datasets with good accuracy and we can use advance deep learning model to get better efficient accuracy.

BIBLIOGRAPHY

1. F.A.Spanhol,L.S.Oliveria, C. Petitjean, L. Heutte, "Breast cancer histopathological Image classification using convolutional neural networks", in: Proceedings of the International Joint Conference on Neural Networks, 2016-October, 2016, pp. 2560

2. Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, Yaman Afadar, Omar Elgendy, "Breast cancer detection using artificial intelligence techniques"in 2021.

3. G. Chugh, S. Kumar, N. Singh, "Survey on machine learning and deep learning app plications in breast cancer diagnosis", Cognit. Comput. 13 (6) (2021) 1451–1470.

4. Zeduo Yuan,Guoming chen,Qiang Chen,Wanyi LiShun Long, "Breast cancer Image Classification Based On CNN Classifier" in 2020.

5. H. Houssein, M.M. Emam, A.A. Ali, P.N. Suganthan, "Deep and machine learning techniques for medical imaging-based breast cancer": a comprehensive review,expert Syst. Appl. 167 (2021) 114161.

6. Moh'd Hadidi ,Abdulsalam Alarabeyyat ,mohannad Alhanahnah, "Breast Cancer Detection Using K-Nearest Neighbor Machine Learning Algorithm" in 2019.

7. A. Aloyayri, A. Krzyzak, " Breast Cancer Classification from Histopathological Im--ages Using Transfer Learning and Deep Neural Networks", in:Lecture Note on the Computer Science (including subseries Lecture Notes in Artificial Intelligence ---- Lecture Notes in Bioinformatics), 12415 LNAI, 2020, pp. 491– 502.

8. José Rouco, Paulo Aguiar, Catarina Eloy, António Polónia, "Classification of breast

cancer histology images using Convolutional Neural Networks" in 2019.

9. Spanhol, F. A., Oliveira, L. S., Cavalin, P. R., Petitjean, C., &Heutte, L. (2017,Oct) "Deep features for breast cancer histopathological image classification".In 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 1868-1873).

10. Yu-Dong Zhang,Chinchun Pan,Xianqing Chen, "Abnormal breast identification by nine-layer convolutional neural network with parametric rectified linear unit and in rank-based stochastic pooling" in 2018.

APPENDIX

 # Importing Libraries

```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=FutureWarning)

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g.
pd.read_csv)
import matplotlib.pyplot as plt
%matplotlib inline

from PIL import Image
from sklearn.model_selection import train_test_split,
StratifiedKFold
from sklearn.utils.class_weight import compute_class_weight
from keras.applications.densenet import DenseNet201
from glob import glob
from skimage.io import imread
from os import listdir

import time
import copy
from tqdm import tqdm_notebook as tqdm


import tensorflow as tf


base_path = "../input/breast-histopathology-
images/IDC_regular_ps50_idx5/"
folder = listdir(base_path)
len(folder)
total_images = 0
for n in range(len(folder)):
    patient_id = folder[n]
    for c in [0, 1]:
        patient_path = base_path + patient_id
        class_path = patient_path + "/" + str(c) + "/"
        subfiles = listdir(class_path)
        total_images += len(subfiles)
print(total_images)


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

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


data.shape


data.target = data.target.astype(np.int)
pos_selection =
np.random.choice(data[data.target==1].index.values, size=50,
replace=False)
neg_selection =
np.random.choice(data[data.target==0].index.values, size=50,
replace=False)


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

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


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

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


def extract_coords(df):
```

```
    coord = df.path.str.rsplit("_", n=4, expand=True)
    coord = coord.drop([0, 1, 4], axis=1)
    coord = coord.rename({2: "x", 3: "y"}, axis=1)
    coord.loc[:, "x"] = coord.loc[:,"x"].str.replace("x", "",
case=False).astype(np.int)
    coord.loc[:, "y"] = coord.loc[:,"y"].str.replace("y", "",
case=False).astype(np.int)
    df.loc[:, "x"] = coord.x.values
    df.loc[:, "y"] = coord.y.values
    return df

def get_cancer_dataframe(patient_id, cancer_id):
    path = base_path + patient_id + "/" + cancer_id
    files = listdir(path)
    dataframe = pd.DataFrame(files, columns=["filename"])
    path_names = path + "/" + dataframe.filename.values
    dataframe = dataframe.filename.str.rsplit("_", n=4,
expand=True)
    dataframe.loc[:, "target"] = np.int(cancer_id)
    dataframe.loc[:, "path"] = path_names
    dataframe = dataframe.drop([0, 1, 4], axis=1)
    dataframe = dataframe.rename({2: "x", 3: "y"}, axis=1)
    dataframe.loc[:, "x"] =
dataframe.loc[:,"x"].str.replace("x", "",
case=False).astype(np.int)
    dataframe.loc[:, "y"] =
dataframe.loc[:,"y"].str.replace("y", "",
case=False).astype(np.int)
    return dataframe

def get_patient_dataframe(patient_id):
    df_0 = get_cancer_dataframe(patient_id, "0")
    df_1 = get_cancer_dataframe(patient_id, "1")
    patient_df = df_0.append(df_1)
    return patient_df


def visualise_breast_tissue_base(patient_id, pred_df=None):
    example_df = get_patient_dataframe(patient_id)
    max_point = [example_df.y.max()-1, example_df.x.max()-1]
    grid = 255*np.ones(shape = (max_point[0] + 50,
max_point[1] + 50, 3)).astype(np.uint8)
    mask = 255*np.ones(shape = (max_point[0] + 50,
max_point[1] + 50, 3)).astype(np.uint8)
    if pred_df is not None:
        patient_df = pred_df[pred_df.patient_id ==
patient_id].copy()
    mask_proba = np.zeros(shape = (max_point[0] + 50,
max_point[1] + 50, 1)).astype(np.float)

    broken_patches = []
```

```
    for n in range(len(example_df)):
        try:
            image = imread(example_df.path.values[n])

            target = example_df.target.values[n]

            x_coord = np.int(example_df.x.values[n])
            y_coord = np.int(example_df.y.values[n])
            x_start = x_coord - 1
            y_start = y_coord - 1
            x_end = x_start + 50
            y_end = y_start + 50

            grid[y_start:y_end, x_start:x_end] = image
            if target == 1:
                mask[y_start:y_end, x_start:x_end, 0] = 250
                mask[y_start:y_end, x_start:x_end, 1] = 0
                mask[y_start:y_end, x_start:x_end, 2] = 0
            if pred_df is not None:

                proba = patient_df[
                    (patient_df.x==x_coord) &
(patient_df.y==y_coord)].proba
                mask_proba[y_start:y_end, x_start:x_end, 0] =
np.float(proba)

        except ValueError:
            broken_patches.append(example_df.path.values[n])


    return grid, mask, broken_patches, mask_proba



def convertNameTrueLabel(num):
    if(num==0):
        return 'True Label = Stage 1 or 2';
    elif(num==1):
        return 'True Label = Stage 3 or 4';
    else:
        return '';
def convertNamePredictedLabel(num):
    if(num==0):
        return 'Predicted Label = Stage 1 or 2';
    elif(num==1):
        return 'Predicted Label = Stage 3 or 4';
    else:
        return '';

def visualise_breast_tissue(patient_id,trueLabel=-
1,predictedLabel=-1):
```

```
    grid, mask, broken_patches,_ =
visualise_breast_tissue_base(patient_id)

    fig, ax = plt.subplots(1,2,figsize=(20,10))
    ax[0].imshow(grid, alpha=0.9)
    ax[1].imshow(mask, alpha=0.8)
    ax[1].imshow(grid, alpha=0.7)
    ax[0].grid(False)
    ax[1].grid(False)
    for m in range(2):
        ax[m].set_xlabel("X-coord")
        ax[m].set_ylabel("Y-coord")
    if(trueLabel!=-1):
        ax[0].set_title("Breast tissue slice of patient: " +
patient_id + '\n' +
convertNameTrueLabel(image_datasets["train"].__getitem__(idx)[
'label']) + '\n' + convertNamePredictedLabel(predictedLabel));
        ax[1].set_title("Cancer tissue colored red \n of
patient: " + patient_id);
    else:
        ax[0].set_title("Breast tissue slice of patient: " +
patient_id);
        ax[1].set_title("Cancer tissue colored red \n of
patient: " + patient_id);


patient_id = "10253"
# get_patient_dataframe(patient_id)
visualise_breast_tissue("10253",100)
visualise_breast_tissue("10254",100)
visualise_breast_tissue("10255",100)
visualise_breast_tissue("10256",100)
visualise_breast_tissue("10257",100)


def visualise_breast_tissue_binary(patient_id):

    fig, ax = plt.subplots(1,1)

    example_df = get_patient_dataframe(patient_id)

    ax.scatter(example_df.x.values, example_df.y.values,
c=example_df.target.values, cmap="coolwarm", s=20);
    ax.set_title("Patient " + patient_id)
    ax.set_xlabel("X coord")
    ax.set_ylabel("Y coord")


#Data preprocessing

data.head()
```

```
data.loc[:, "target"] = data.target.astype(np.str)
data.info()

patients = data.patient_id.unique()

train_ids, sub_test_ids = train_test_split(patients,
                                           test_size=0.3,
                                           random_state=0)
test_ids, dev_ids = train_test_split(sub_test_ids,
test_size=0.5, random_state=0)


train_df = data.loc[data.patient_id.isin(train_ids),:].copy()
test_df = data.loc[data.patient_id.isin(test_ids),:].copy()
dev_df = data.loc[data.patient_id.isin(dev_ids),:].copy()

train_df = extract_coords(train_df)
test_df = extract_coords(test_df)
dev_df = extract_coords(dev_df)


from torch.utils.data import Dataset
class BreastCancerDataset(Dataset):

    def __init__(self, df):
        self.states = df

    def __len__(self):
        return len(self.states)

    def __getitem__(self, idx):
        patient_id = self.states.patient_id.values[idx]
        x_coord = self.states.x.values[idx]
        y_coord = self.states.y.values[idx]
        image_path = self.states.path.values[idx]
        image = Image.open(image_path)
        image = image.convert('RGB')

        if "target" in self.states.columns.values:
            target = np.int(self.states.target.values[idx])
        else:
            target = None

        return {"image": image,
                "label": target,
                "patient_id": patient_id,
                "x": x_coord,
                "y": y_coord}


train_dataset = BreastCancerDataset(train_df)
```

```
dev_dataset = BreastCancerDataset(dev_df)
test_dataset = BreastCancerDataset(test_df)


image_datasets = {"train": train_dataset, "dev": dev_dataset,
"test": test_dataset}
dataset_sizes = {x: len(image_datasets[x]) for x in ["train",
"dev", "test"]}


patient_id = "10253"
# get_patient_dataframe(patient_id)
visualise_breast_tissue_binary(patient_id)


print("Training Dataset : ",dataset_sizes["train"])
print("Dev Dataset : ",dataset_sizes["dev"])
print("Test Dataset : ",dataset_sizes["test"])


plt.imshow(image_datasets["test"].__getitem__(0)['image'])


#Creating database suitable for tensorflow
train_images = []
train_labels = []
# for i in range(dataset_sizes["train"]):
for i in range(1000):

train_images.append(np.array(image_datasets["train"].__getitem
__(i)['image']))

train_labels.append(image_datasets["train"].__getitem__(i)['la
bel'])

train_images = np.array(train_images)
train_labels = np.array(train_labels)


train_images.shape


#Creating database suitable for tensorflow
test_images = []
test_labels = []
# for i in range(dataset_sizes["test"]):
for i in range(1000,2000):

test_images.append(np.array(image_datasets["train"].__getitem_
_(i)['image']))
```

```
test_labels.append(image_datasets["train"].__getitem__(i)['lab
el'])

test_images = np.array(test_images)
test_labels = np.array(test_labels)


test_images.shape


#Training
BATCH_SIZE = 32
NUM_CLASSES = 2

from tensorflow.keras import datasets, layers, models

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images
/ 255.0


model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(50, 50, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(2))


model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
=True),
              metrics=['accuracy'])
densenet = DenseNet201(
    weights = 'imagenet',
    include_top = False,
    input_shape=(50, 50, 3)
)
model = build_model(densenet,lr=1e-4)
history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images,
test_labels))
```

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label =
'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(test_images,
test_labels, verbose=2)

print("#ACCURACY")
print(test_acc)

#image predictions
def predict_single_image(idx):
    img = []

img.append(np.array(image_datasets["train"].__getitem__(idx)['
image']))
    img = np.array(img)
    print("Stage1 or 2:1","Stage 3 or 4 : 0")
    print("For sample number " + str(idx))
    print("True Lable :
",image_datasets["train"].__getitem__(idx)['label'])
    print("Predicted Label : ",
model.predict(img).argmax(axis=-1)[0])
    visualise_breast_tissue(str(10253+idx-
25),image_datasets["train"].__getitem__(idx)['label'],model.pr
edict(img).argmax(axis=-1)[0])



print("#SAMPLE OUTPUT")
predict_single_image(25)
predict_single_image(26)
predict_single_image(27)
predict_single_image(28)
```