# PROJECT REPORT ON

# DEEP LEARNING TECHNIQUES FOR BREAST CANCER PREDICTION USING PYTHON

**1. INTRODUCTION :**

Breast cancer is one of the main causes of cancer deaths worldwide and is the second most common cancer in women and men worldwide. In 2010, it represented about 12 percent of all new cancer cases and 25 percent of all cancers in women while a decade later, the figures have augmented to more than 75% and 89%, respectively.

**1.1 OVERVIEW :**

Computer-aided diagnosis provides a most important option for image diagnosis, which can improve the reliability of experts’ decision-making. Automatic and precision classification for breast cancer histopathological image is of great importance in clinical application for identifying the malignant stage from histopathological images.

**1.2 PURPOSE :**

Computer-aided diagnosis systems showed potential for improving diagnostic accuracy. Since early detection and prevention can significantly reduce the chances of death, it is important to detect breast cancer as early as possible. This project is developed to predict the cancer at the earliest stage using Convolutional Neural Networks from the concept of Deep-Learning.

**2. LITERATURE SURVEY :**

To look at how we have come across this project overcoming the existing problem, we look at these aspects with the proposed solution.

**2.1 EXISTING PROBLEM :**

Breast cancer is typically detected either during screening, before symptoms have developed, or after a woman notices a lump. There are various ways to detect breast cancer including Mammography, Magnetic Resonance Imaging (MRI) Scans, Computed Tomography (CT) Scans, Ultrasound, and Nuclear Imaging. When cancer is suspected, tissue for microscopic analysis is usually obtained from a needle and is stained with H&E, Hematoxylin and Eosin. Selection of the biopsy type is based on multiple factors, including the size, location of the mass, patient factors, preferences and resources. But all of them are very expensive, less accurate and time-taking. Sometimes, it even takes months together to get the final reports and all the aforementioned technologies have their own shortcomings.

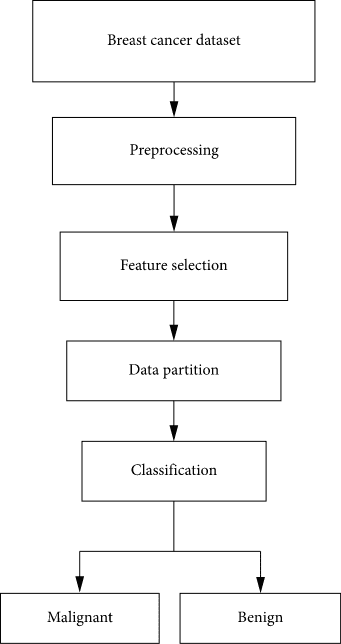
**2.2 PROPOSED SOLUTION :**

I have proposed a solution where we build an algorithm to automatically identify whether a patient is suffering from breast cancer or not by looking at biopsy images in less than a minute. This algorithm had to be extremely accurate because lives of people are at stake and time was a real concern. This project results the output in two categories, which are benign and malignant, which mean non cancerous and cancerous, by displaying benign and malignant.

**3. TECHNICAL ANALYSIS :**

**3.1 BLOCK DIAGRAM :**

In the below attached diagram, we see images of biopsy taken as the inputs. We then process them using packages like numpy, ImageDataGenerator, matplotlib, keras etc and classify them from many layers so that the test data gives us accurate results based on training data.



**3.2 HARDWARE AND SOFTWARE DESIGNING :**

**Hardware:**

* Manufacture : MicrosoftCorporation
* Processor : Intel@Core i7
* Installed memory (RAM) : 8 GB System
* Type : 64-bit Operating System, x64-based Processor

**Software:**

* Operating System : Window10pro
* Anaconda version : 3.8 for Windows 8.1
* Python version : 3.8
* Tensorflow version : 1.14.0 usinganaconda
* Packages : Keras, Matplotlib, Pandas, Numpy etc.

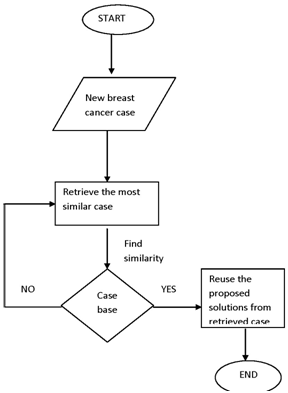
We have to install all the needed packages before starting the project.

Download the zipped files and unzip them in a directory.Now, inside the inner breast cancer classification directory, create directory datasets, inside this create directory original. Download the dataset. Run the model.

**4. EXPERIMENTAL INVESTIGATIONS :**

1. I have proposed an automatic breast cancer detection technique that gives prediction accuracy of around 90% for the true class.
2. I have used histopathology images from biopsy, in which tissues affected by the tumor are extracted and stained with H & E.
3. CNN is used for feature extraction, and classification is done using all the various packages to get accurate and quick results.
4. The result is in the form of Binary Classification between two classes of cancer. Benign is class 0 and malignant is class 1.

**5. FLOWCHART :**



**6. RESULT :**

In this project, we designed a new Convolutional Neural Network, the Breast Cancer Histopathology Image Classification Network (BHCNet), for the classification of breast cancer histopathology images. We built a classifier which uses 80% as training dataset, of which 10% is for validation dataset and the rest 20% for test dataset. We designed a small SE-ResNet(Squeeze and Excitation) module with fewer parameters to reduce the training parameters of the model, and to reduce the risk of model over-fitting.

**7. ADVANTAGES AND DISADVANTAGES :**

**ADVANTAGES :**

1. This project helps patients get to know their cancer status really quick.
2. This project will help people to no longer spend thousands and thousands of bugs for getting their reports. This is very economical.
3. This project uses CNN techniques and fetches the results without any physical risk or stain identifications. There are also no or ignorable shortcomings in the project developed so far.
4. Data input can be from various resources.

**DISADVANTAGES :**

1. We have designed this project in Python and is executable only in a properly fixed environment, fulfilling all the hardware and software requirements.
2. It requires high speed data connectivity in real-life situations. Therefore, since the dataset used is large, we need to make sure that the internet connectivity is high and the run-time environment is user-friendly.

**8. APPLICATIONS :**

The developed system can be used in hospitals,diagnostic centres, implantation centers, laboratories etc where people can go and find out if they are affected with breast cancers or for any other purpose. This is a very practical implementation of identifying breast cancer in patients and is a very feasible and economical method.

**9. CONCLUSION :**

I have developed this project where we can identify if the cancer developed is benign or malignant. We do this by taking the images of biopsies and processing them through various layers of CNN and obtaining the results in either of the two cases(0 or 1). Atlast, we also use a confusion matrix to derive the performance of the model.

**10. FUTURE SCOPE :**

1. This project can be used in all the hospitals to identify the situation of patients’ body and recover the malignant patients at the early stages.
2. The project can be used to further add-on other features to improvise the project and get higher efficiency with multiple facilities.
3. The use of Deep-Learning techniques is better than using density based models to get accurate results.

**11. BIBLIOGRAPHY :**

1. <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0214587>
2. <https://data-flair.training/blogs/project-in-python-breast-cancer-classification/>
3. <https://towardsdatascience.com/convolutional-neural-network-for-breast-cancer-classification-52f1213dcc9>

**12. APPENDIX :**

Source code is as follows:

1. Config.py :

1. import os

2.

3. INPUT\_DATASET = "datasets/original"

4.

5. BASE\_PATH = "datasets/idc"

6. TRAIN\_PATH = os.path.sep.join([BASE\_PATH, "training"])

7. VAL\_PATH = os.path.sep.join([BASE\_PATH, "validation"])

8. TEST\_PATH = os.path.sep.join([BASE\_PATH, "testing"])

9.

10. TRAIN\_SPLIT = 0.8

11. VAL\_SPLIT = 0.1

1. build\_dataset.py :

1. from casncernet import config

2. from imutils import paths

3. import random, shutil, os

4.

5. originalPaths=list(paths.list\_images(config.INPUT\_DATASET))

6. random.seed(7)

7. random.shuffle(originalPaths)

8.

9. index=int(len(originalPaths)\*config.TRAIN\_SPLIT)

10. trainPaths=originalPaths[:index]

11. testPaths=originalPaths[index:]

12.

13. index=int(len(trainPaths)\*config.VAL\_SPLIT)

14. valPaths=trainPaths[:index]

15. trainPaths=trainPaths[index:]

16.

17. datasets=[("training", trainPaths, config.TRAIN\_PATH),

18. ("validation", valPaths, config.VAL\_PATH),

19. ("testing", testPaths, config.TEST\_PATH)

20. ]

21.

22. for (setType, originalPaths, basePath) in datasets:

23. print(f'Building {setType} set')

24.

25. if not os.path.exists(basePath):

26. print(f'Building directory {base\_path}')

27. os.makedirs(basePath)

28.

29. for path in originalPaths:

30. file=path.split(os.path.sep)[-1]

31. label=file[-5:-4]

32.

33. labelPath=os.path.sep.join([basePath,label])

34. if not os.path.exists(labelPath):

35. print(f'Building directory {labelPath}')

36. os.makedirs(labelPath)

37.

38. newPath=os.path.sep.join([labelPath, file])

39. shutil.copy2(inputPath, newPath)

1. cancernet.py :

1. from keras.models import Sequential

2. from keras.layers.normalization import BatchNormalization

3. from keras.layers.convolutional import SeparableConv2D

4. from keras.layers.convolutional import MaxPooling2D

5. from keras.layers.core import Activation

6. from keras.layers.core import Flatten

7. from keras.layers.core import Dropout

8. from keras.layers.core import Dense

9. from keras import backend as K

10.

11. class CancerNet:

12. @staticmethod

13. def build(width,height,depth,classes):

14. model=Sequential()

15. shape=(height,width,depth)

16. channelDim=-1

17.

18. if K.image\_data\_format()=="channels\_first":

19. shape=(depth,height,width)

20. channelDim=1

21.

22. model.add(SeparableConv2D(32, (3,3), padding="same",input\_shape=shape))

23. model.add(Activation("relu"))

24. model.add(BatchNormalization(axis=channelDim))

25. model.add(MaxPooling2D(pool\_size=(2,2)))

26. model.add(Dropout(0.25))

27.

28. model.add(SeparableConv2D(64, (3,3), padding="same"))

29. model.add(Activation("relu"))

30. model.add(BatchNormalization(axis=channelDim))

31. model.add(SeparableConv2D(64, (3,3), padding="same"))

32. model.add(Activation("relu"))

33. model.add(BatchNormalization(axis=channelDim))

34. model.add(MaxPooling2D(pool\_size=(2,2)))

35. model.add(Dropout(0.25))

36.

37. model.add(SeparableConv2D(128, (3,3), padding="same"))

38. model.add(Activation("relu"))

39. model.add(BatchNormalization(axis=channelDim))

40. model.add(SeparableConv2D(128, (3,3), padding="same"))

41. model.add(Activation("relu"))

42. model.add(BatchNormalization(axis=channelDim))

43. model.add(SeparableConv2D(128, (3,3), padding="same"))

44. model.add(Activation("relu"))

45. model.add(BatchNormalization(axis=channelDim))

46. model.add(MaxPooling2D(pool\_size=(2,2)))

47. model.add(Dropout(0.25))

48.

49. model.add(Flatten())

50. model.add(Dense(256))

51. model.add(Activation("relu"))

52. model.add(BatchNormalization())

53. model.add(Dropout(0.5))

54.

55. model.add(Dense(classes))

56. model.add(Activation("softmax"))

57.

58. return model

1. train\_model.py :

1. import matplotlib

2. matplotlib.use("Agg")

3.

4. from keras.preprocessing.image import ImageDataGenerator

5. from keras.callbacks import LearningRateScheduler

6. from keras.optimizers import Adagrad

7. from keras.utils import np\_utils

8. from sklearn.metrics import classification\_report

9. from sklearn.metrics import confusion\_matrix

10. from cancernet.cancernet import CancerNet

11. from cancernet import config

12. from imutils import paths

13. import matplotlib.pyplot as plt

14. import numpy as np

15. import os

16.

17. NUM\_EPOCHS=40; INIT\_LR=1e-2; BS=32

18.

19. trainPaths=list(paths.list\_images(config.TRAIN\_PATH))

20. lenTrain=len(trainPaths)

21. lenVal=len(list(paths.list\_images(config.VAL\_PATH)))

22. lenTest=len(list(paths.list\_images(config.TEST\_PATH)))

23.

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

25. trainLabels=np\_utils.to\_categorical(trainLabels)

26. classTotals=trainLabels.sum(axis=0)

27. classWeight=classTotals.max()/classTotals

28.

29. trainAug = ImageDataGenerator(

30. rescale=1/255.0,

31. rotation\_range=20,

32. zoom\_range=0.05,

33. width\_shift\_range=0.1,

34. height\_shift\_range=0.1,

35. shear\_range=0.05,

36. horizontal\_flip=True,

37. vertical\_flip=True,

38. fill\_mode="nearest")

39.

40. valAug=ImageDataGenerator(rescale=1 / 255.0)

41.

42. trainGen = trainAug.flow\_from\_directory(

43. config.TRAIN\_PATH,

44. class\_mode="categorical",

45. target\_size=(48,48),

46. color\_mode="rgb",

47. shuffle=True,

48. batch\_size=BS)

49. valGen = valAug.flow\_from\_directory(

50. config.VAL\_PATH,

51. class\_mode="categorical",

52. target\_size=(48,48),

53. color\_mode="rgb",

54. shuffle=False,

55. batch\_size=BS)

56. testGen = valAug.flow\_from\_directory(

57. config.TEST\_PATH,

58. class\_mode="categorical",

59. target\_size=(48,48),

60. color\_mode="rgb",

61. shuffle=False,

62. batch\_size=BS)

63.

64. model=CancerNet.build(width=48,height=48,depth=3,classes=2)

65. opt=Adagrad(lr=INIT\_LR,decay=INIT\_LR/NUM\_EPOCHS)

66. model.compile(loss="binary\_crossentropy",optimizer=opt,metrics=["accuracy"])

67.

68.

69. M=model.fit\_generator(

70. trainGen,

71. steps\_per\_epoch=lenTrain//BS,

72. validation\_data=valGen,

73. validation\_steps=lenVal//BS,

74. class\_weight=classWeight,

75. epochs=NUM\_EPOCHS)

76.

77. print("Now evaluating the model")

78. testGen.reset()

79. pred\_indices=model.predict\_generator(testGen,steps=(lenTest//BS)+1)

80.

81. pred\_indices=np.argmax(pred\_indices,axis=1)

82.

83. print(classification\_report(testGen.classes, pred\_indices, target\_names=testGen.class\_indices.keys()))

84.

85. cm=confusion\_matrix(testGen.classes,pred\_indices)

86. total=sum(sum(cm))

87. accuracy=(cm[0,0]+cm[1,1])/total

88. specificity=cm[1,1]/(cm[1,0]+cm[1,1])

89. sensitivity=cm[0,0]/(cm[0,0]+cm[0,1])

90. print(cm)

91. print(f'Accuracy: {accuracy}')

92. print(f'Specificity: {specificity}')

93. print(f'Sensitivity: {sensitivity}')

94.

95. N = NUM\_EPOCHS

96. plt.style.use("ggplot")

97. plt.figure()

98. plt.plot(np.arange(0,N), M.history["loss"], label="train\_loss")

99. plt.plot(np.arange(0,N), M.history["val\_loss"], label="val\_loss")

100. plt.plot(np.arange(0,N), M.history["acc"], label="train\_acc")

101. plt.plot(np.arange(0,N), M.history["val\_acc"], label="val\_acc")

102. plt.title("Training Loss and Accuracy on the IDC Dataset")

103. plt.xlabel("Epoch No.")

104. plt.ylabel("Loss/Accuracy")

105. plt.legend(loc="lower left")

106. plt.savefig('plot.png')

5 .App.py :

* 1. from \_\_future\_\_ import division, print\_function
  2. # coding=utf-8
  3. import sys
  4. import os
  5. import glob
  6. import numpy as np
  7. from keras.preprocessing import image
  8. from keras.applications.imagenet\_utils import preprocess\_input, decode\_predictions
  9. from keras.models import load\_model
  10. from keras import backend
  11. from tensorflow.keras import backend
  12. import tensorflow as tf
  13. global graph
  14. graph=tf.compat.v1.get\_default\_graph()
  15. #global graph
  16. #graph = tf.get\_default\_graph()
  17. from skimage.transform import resize
  18. # Flask utils
  19. from flask import Flask, redirect, url\_for, request, render\_template
  20. from werkzeug.utils import secure\_filename
  21. from gevent.pywsgi import WSGIServer
  22. # Define a flask app
  23. app = Flask(\_\_name\_\_)
  24. # Model saved with Keras model.save()
  25. MODEL\_PATH = 'Breast\_cancer.h5'
  26. # Load your trained model
  27. model = load\_model(MODEL\_PATH)
  28. # Necessary
  29. # print('Model loaded. Start serving...')
  30. # You can also use pretrained model from Keras
  31. # Check https://keras.io/applications/
  32. #from keras.applications.resnet50 import ResNet50
  33. #model = ResNet50(weights='imagenet')
  34. #model.save('')
  35. print('Model loaded. Check http://127.0.0.1:5000/')
  36. @app.route('/', methods=['GET'])
  37. def index():
  38. # Main page
  39. return render\_template('index.html')
  40. @app.route('/predict', methods=['GET', 'POST'])
  41. def upload():
  42. if request.method == 'POST':
  43. # Get the file from post request
  44. f = request.files['file']
  45. # Save the file to ./uploads
  46. basepath = os.path.dirname(\_\_file\_\_)
  47. file\_path = os.path.join(
  48. basepath, 'uploads', secure\_filename(f.filename))
  49. f.save(file\_path)
  50. img = image.load\_img(file\_path, target\_size=(48, 48))
  51. x = image.img\_to\_array(img)
  52. x = np.expand\_dims(x, axis=0)
  53. with graph.as\_default():
  54. preds = model.predict\_classes(x)
  55. print("prediction ",preds)
  56. index = ['BENIGN','MALIGNANT']
  57. text = "Cancer Type: "+index[preds[0]]
  58. # ImageNet Decode
  59. return text
  60. if \_\_name\_\_ == '\_\_main\_\_':
  61. app.run(debug=False,threaded = False)

The **html and JS files** designed for the **User Interface** would be as follows :

1.base.html :

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Cancer Prediction</title>

<link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">

<script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>

<script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>

<script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>

<link href="{{ url\_for('static', filename='css/main.css') }}" rel="stylesheet">

<style>

head{

color : rgb(196, 25, 125) ;

text-align:center;

}

body{

background-color: #FAF127 ;

text-align:center;

color : rgb(196, 25, 125) ;

}

h1{

color: #060400;

text-align:center;

}

p,a{

font-family: verdana;

font-size: 20px;

}

.button {

background-color: yellow;

border: none;

color: white;

padding: 15px 32px;

text-align: center;

cursor: pointer;}

</style>

</head>

<body>

<nav class="navbar navbar-dark bg-dark">

<div class="container">

<a class="navbar-brand" text-align=center href="#"> ~ CANCER PREDICTION ~ </a>

<button class="btn btn-outline-secondary my-2 my-sm-0" type="submit">Help</button>

</div>

</nav>

<div class="container">

<div id="content" style="margin-top:2em">{% block content %}{% endblock %}</div>

</div>

</body>

<footer>

<script src="{{ url\_for('static', filename='js/main.js') }}" type="text/javascript"></script>

</footer>

</html>

2.Index.html :

{% extends "base.html" %} {% block content %}

<style>

div {

text-align: center;

font-family: verdana;

font-size: 20px;

}

h2{

text-align:center;

}

</style>

<h2 style="color:black;"> Welcome to Breast Cancer Risk Prediction</h2>

<div><center>

<form id="upload-file" method="post" enctype="multipart/form-data">

<label for="imageUpload" class="upload-label">

Choose a photo from Device

</label> <br> </br>

<input type="file" name="file" id="imageUpload" accept=".png, .jpg, .jpeg">

</form>

<div class="image-section" style="display:none;">

<div class="img-preview">

<div id="imagePreview">

</div>

</div>

<div><center>

<button type="button" class="btn btn-primary btn-lg " id="btn-predict">Identify</button>

</center> </div>

</div>

<div class="loader" style="display:none;"></div>

<h3 id="result">

<span> </span>

</h3>

</center>

</div>

{% endblock %}

3.Main. css:

.img-preview {

width: 256px;

height: 256px;

position: relative;

border: 5px solid #F8F8F8;

box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);

margin-top: 1em;

margin-bottom: 1em;

}

.img-preview>div {

width: 100%;

height: 100%;

background-size: 256px 256px;

background-repeat: no-repeat;

background-position: center;

}

input[type="file"] {

display: none;

}

.upload-label{

display: inline-block;

padding: 12px 30px;

background: #39D2B4;

color: #fff;

font-size: 1em;

transition: all .4s;

cursor: pointer;

}

.upload-label:hover{

background: #34495E;

color: #39D2B4;

}

.loader {

border: 8px solid #f3f3f3; /\* Light grey \*/

border-top: 8px solid #3498db; /\* Blue \*/

border-radius: 50%;

width: 50px;

height: 50px;

animation: spin 1s linear infinite;

}

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

4. Main. js:

$(document).ready(function () {

// Init

$('.image-section').hide();

$('.loader').hide();

$('#result').hide();

// Upload Preview

function readURL(input) {

if (input.files && input.files[0]) {

var reader = new FileReader();

reader.onload = function (e) {

$('#imagePreview').css('background-image', 'url(' + e.target.result + ')');

$('#imagePreview').hide();

$('#imagePreview').fadeIn(650);

}

reader.readAsDataURL(input.files[0]);

}

}

$("#imageUpload").change(function () {

$('.image-section').show();

$('#btn-predict').show();

$('#result').text('');

$('#result').hide();

readURL(this);

});

// Predict

$('#btn-predict').click(function () {

var form\_data = new FormData($('#upload-file')[0]);

// Show loading animation

$(this).hide();

$('.loader').show();

// Make prediction by calling api /predict

$.ajax({

type: 'POST',

url: '/predict',

data: form\_data,

contentType: false,

cache: false,

processData: false,

async: true,

success: function (data) {

// Get and display the result

$('.loader').hide();

$('#result').fadeIn(600);

$('#result').text(' Result: ' + data);

console.log('Success!');

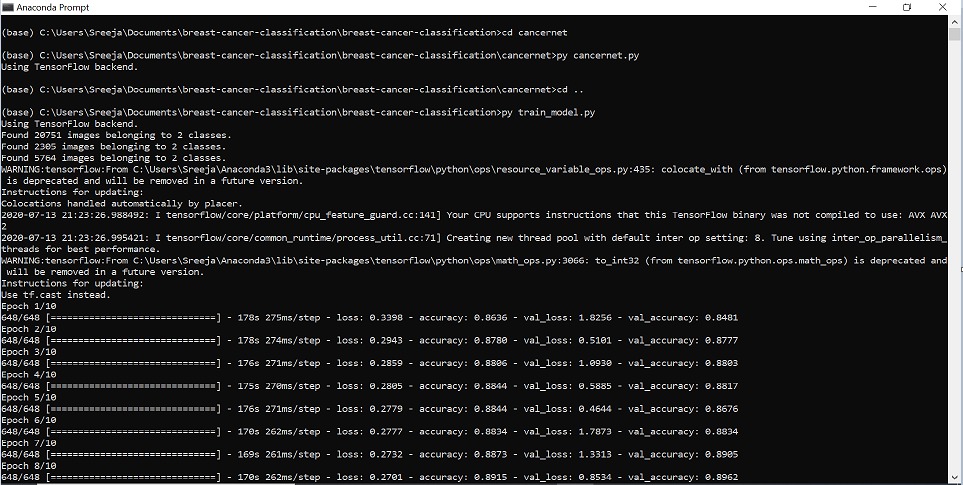
},

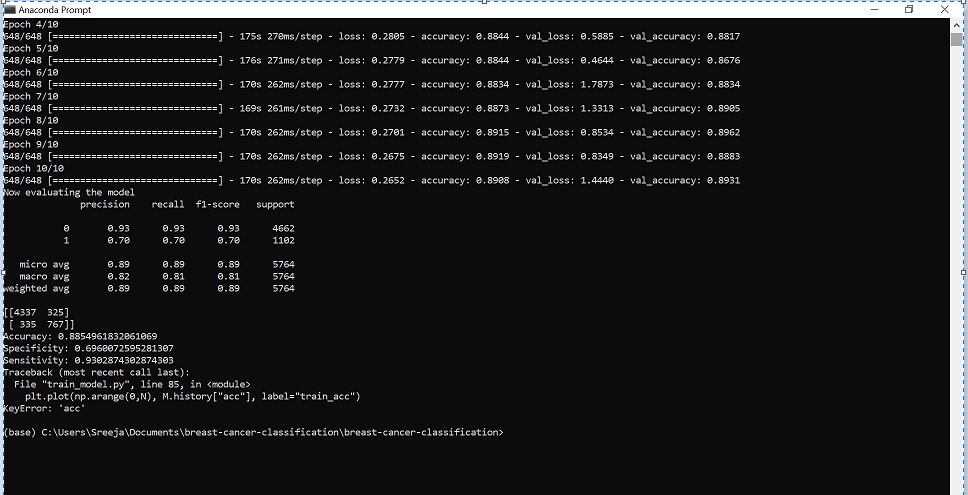
});

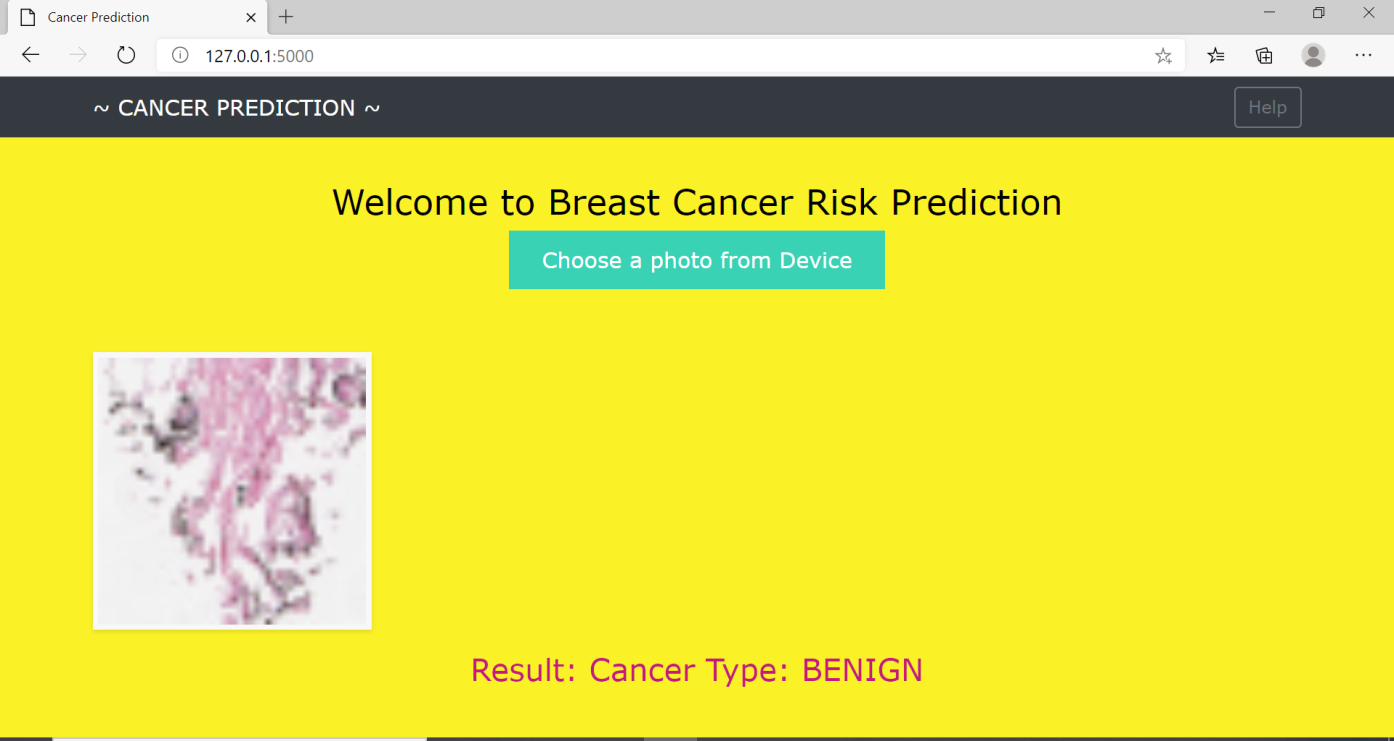
});

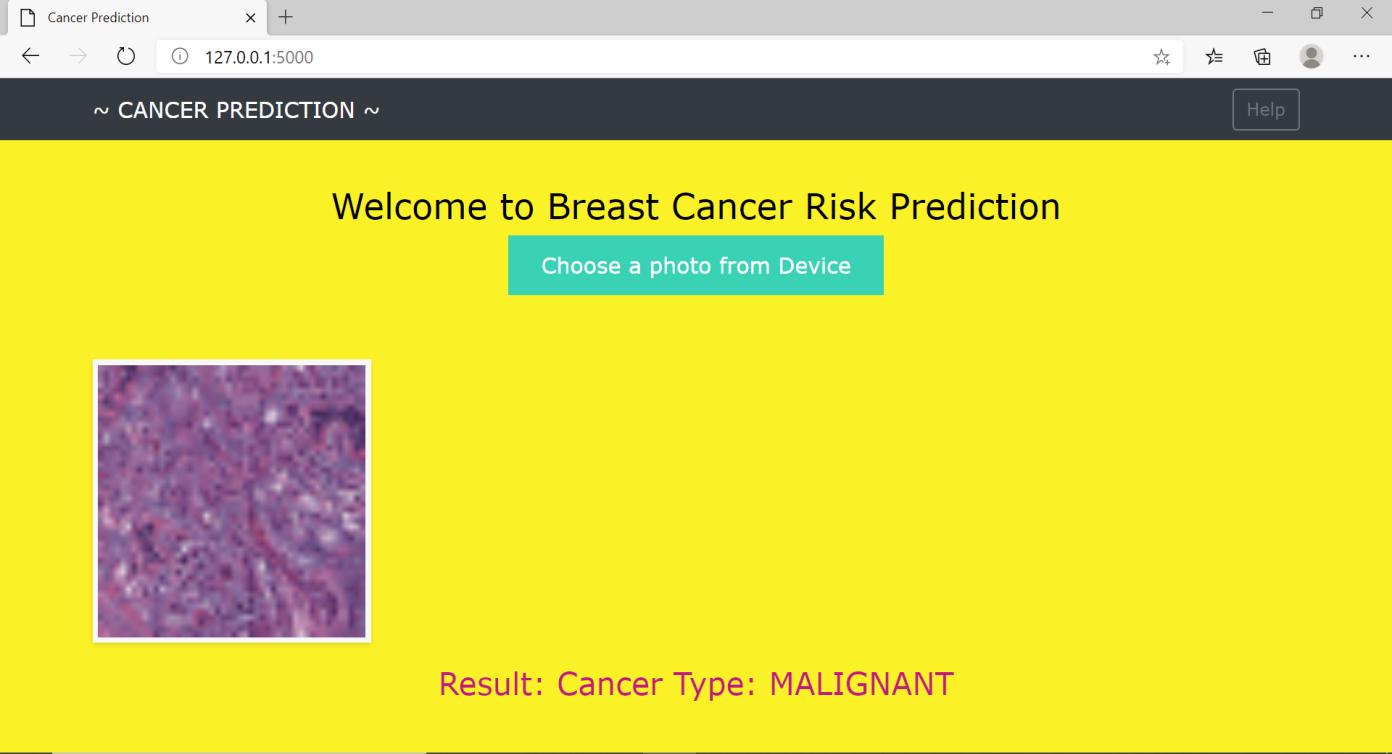
});

**OUTPUT SCREENSHOTS :**









**Developed by :**

Name : Harshita Sanjay Gaddamwar

Internship Title : RSIP Career Basic AI 064

Project ID : SPS\_PRO\_202

Project Title : Deep Learning Techniques for Breast Cancer Risk Prediction using Python

# --- THANK YOU ---