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.

<u>2.1 EXISTING PROBLEM :</u>

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.

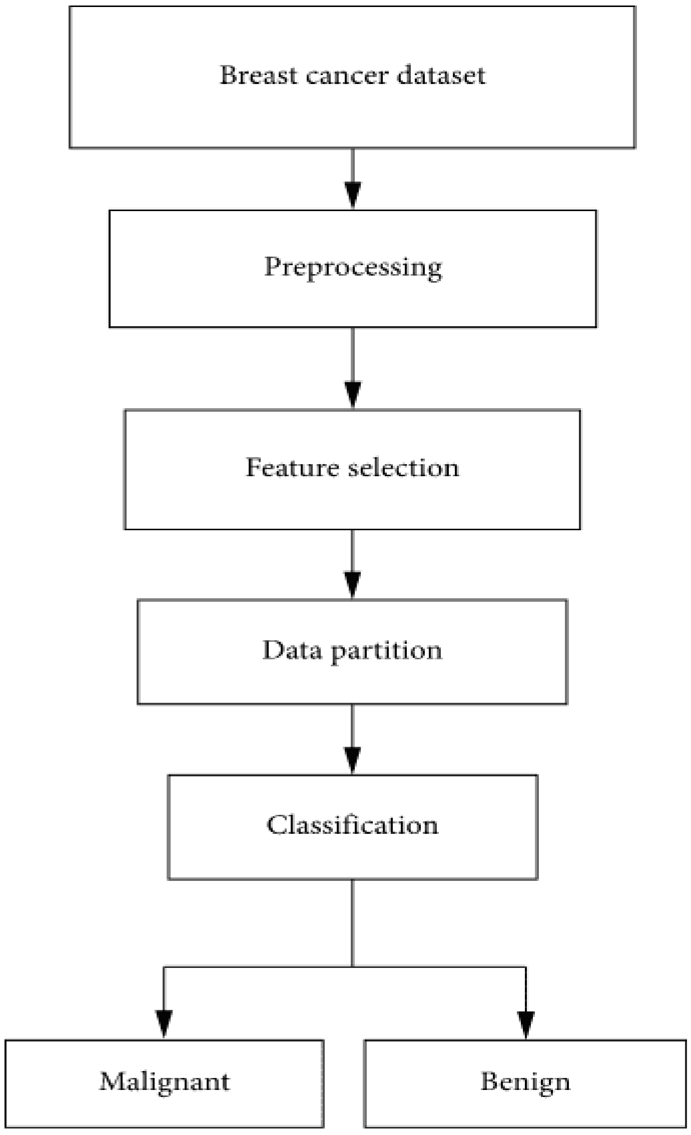
<u>2.2 PROPOSED SOLUTION :</u>

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:

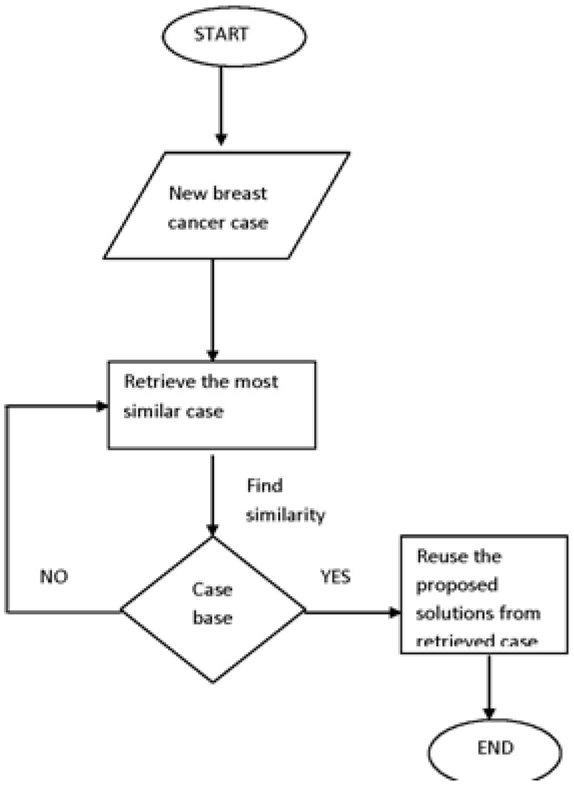
➤ Operating System	: Window10pro
➤ Anaconda version	: 3.8 for Windows 8.1
➤ Python version	: 3.8
➤ Tensorflow version	: 1.14.0 usinganaconda
➤ Packages	: Keras, Matplotlib, Pandas, Numpy etc.

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.

<u>5. FLOWCHART :</u>


START
New breast cancer case
Retrieve the most similar case
Find similarity
NO
Case base
YES
Reuse the proposed solutions from retrieved case
END

<u>6. RESULT :</u>

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 :

```
import os

INPUT_DATASET = "datasets/original"

BASE_PATH = "datasets/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 = 0.1
```

2. build_dataset.py :

```
from casncernet import config
from imutils import paths
import random, shutil, os

originalPaths=list(paths.list_images(config.INPUT_DATASET))
random.seed(7)
random.shuffle(originalPaths)

index=int(len(originalPaths)*config.TRAIN_SPLIT)
trainPaths=originalPaths[:index]
testPaths=originalPaths[index:]

index=int(len(trainPaths)*config.VAL_SPLIT)
valPaths=trainPaths[:index]
trainPaths=trainPaths[index:]

datasets=[("training", trainPaths, config.TRAIN_PATH),
("validation", valPaths, config.VAL_PATH),
("testing", testPaths, config.TEST_PATH)
]

for (setType, originalPaths, basePath) in datasets:
print(f'Building {setType} set')

if not os.path.exists(basePath):
print(f'Building directory {base_path}')
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(inputPath, newPath)
```

3. cancernet.py :

```
from keras.models import Sequential
from keras.layers.normalization 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

class CancerNet:
@staticmethod
def build(width,height,depth,classes):
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"))

return model
```

4. train_model.py :

```
import matplotlib
matplotlib.use("Agg")

from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import LearningRateScheduler
from keras.optimizers import Adagrad
from keras.utils import np_utils
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from cancernet.cancernet import CancerNet
from cancernet import config
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os

NUM_EPOCHS=40; INIT_LR=1e-2; BS=32

trainPaths=list(paths.list_images(config.TRAIN_PATH))
lenTrain=len(trainPaths)
lenVal=len(list(paths.list_images(config.VAL_PATH)))
lenTest=len(list(paths.list_images(config.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=classTotals.max()/classTotals

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(
config.TRAIN_PATH,
class_mode="categorical",
target_size=(48,48),
color_mode="rgb",
shuffle=True,
batch_size=BS)
valGen = valAug.flow_from_directory(
config.VAL_PATH,
class_mode="categorical",
target_size=(48,48),
color_mode="rgb",
shuffle=False,
batch_size=BS)
testGen = valAug.flow_from_directory(
config.TEST_PATH,
class_mode="categorical",
target_size=(48,48),
color_mode="rgb",
shuffle=False,
batch_size=BS)

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

```

print("Now evaluating the model")
testGen.reset()
pred_indices=model.predict_generator(testGen,steps=(lenTest//BS)+1)

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

print(classification_report(testGen.classes, pred_indices, target_names=testGen.class_indices.keys()))

cm=confusion_matrix(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
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0,N), M.history["loss"], label="train_loss")
plt.plot(np.arange(0,N), M.history["val_loss"], label="val_loss")
plt.plot(np.arange(0,N), M.history["acc"], label="train_acc")
plt.plot(np.arange(0,N), M.history["val_acc"], label="val_acc")
plt.title("Training Loss and Accuracy on the IDC Dataset")
plt.xlabel("Epoch No.")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig('plot.png')
```

5 .App.py :

```
from __future__ import division, print_function
# coding=utf-8
import sys
import os
import glob
import numpy as np
from keras.preprocessing import image
from keras.applications.imagenet_utils import preprocess_input, decode_predictions
from keras.models import load_model
from keras import backend
from tensorflow.keras import backend

import tensorflow as tf

global graph
graph=tf.compat.v1.get_default_graph()

#global graph
#graph = tf.get_default_graph()


from skimage.transform import resize

# Flask utils
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer

# Define a flask app
app = Flask(__name__)

# Model saved with Keras model.save()
MODEL_PATH = 'Breast_cancer.h5'

# Load your trained model
model = load_model(MODEL_PATH)
       # Necessary
# print('Model loaded. Start serving...')

# You can also use pretrained model from Keras
# Check https://keras.io/applications/
#from keras.applications.resnet50 import ResNet50
#model = ResNet50(weights='imagenet')
#model.save('')
print('Model loaded. Check http://127.0.0.1:5000/')
@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['file']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        img = image.load_img(file_path, target_size=(48, 48))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)

        with graph.as_default():
            preds = model.predict_classes(x)
            print("prediction  ",preds)
        index = ['BENIGN','MALIGNANT']
        text = "Cancer Type: "+index[preds[0]]
               # ImageNet Decode
        return text
if __name__ == '__main__':
    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 %}
```

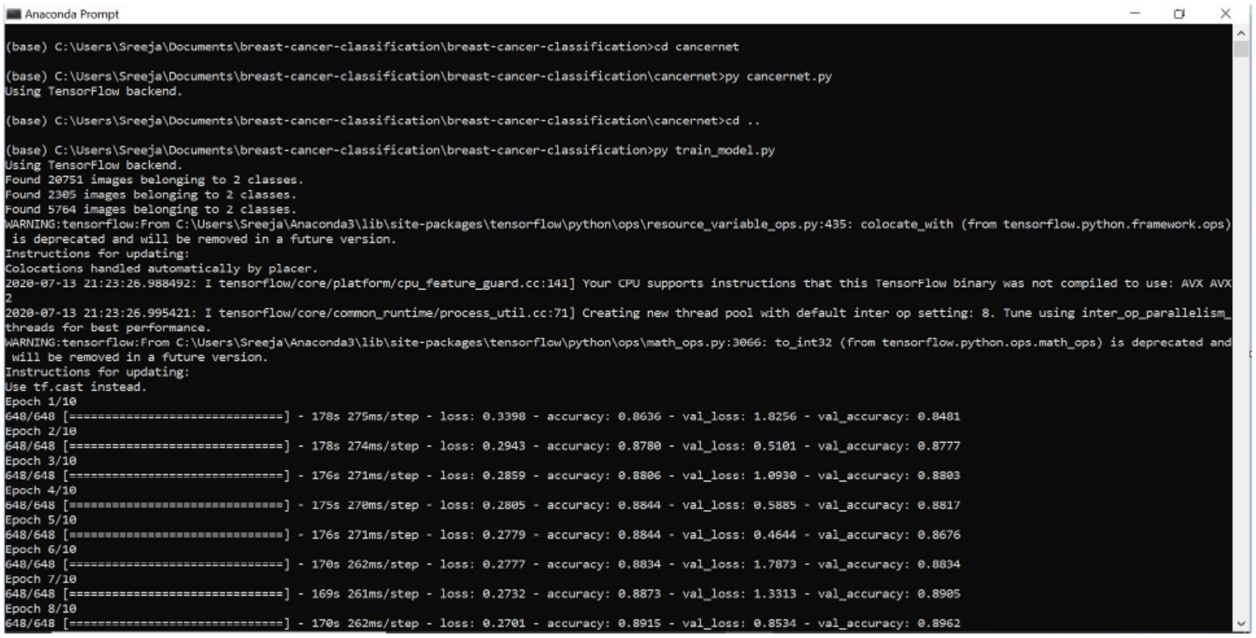
3.Main. css:

```
.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); }
}
```

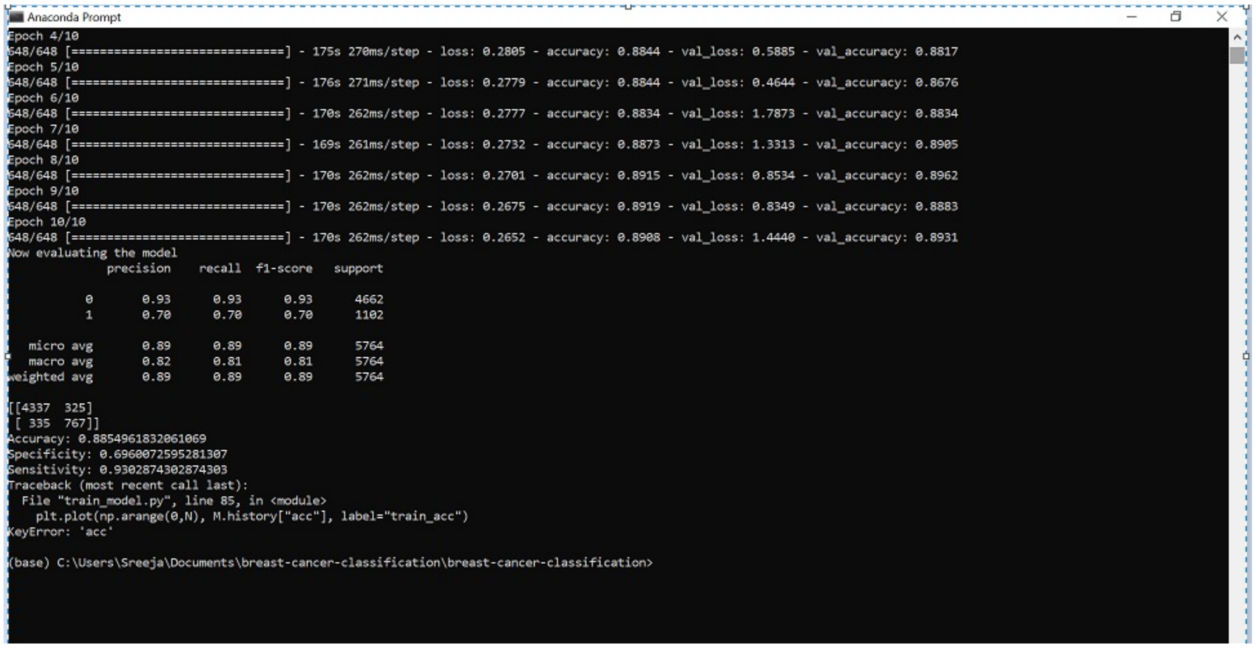
4. Main. js:

```
     $(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!');
            },
        });
    });
});
```

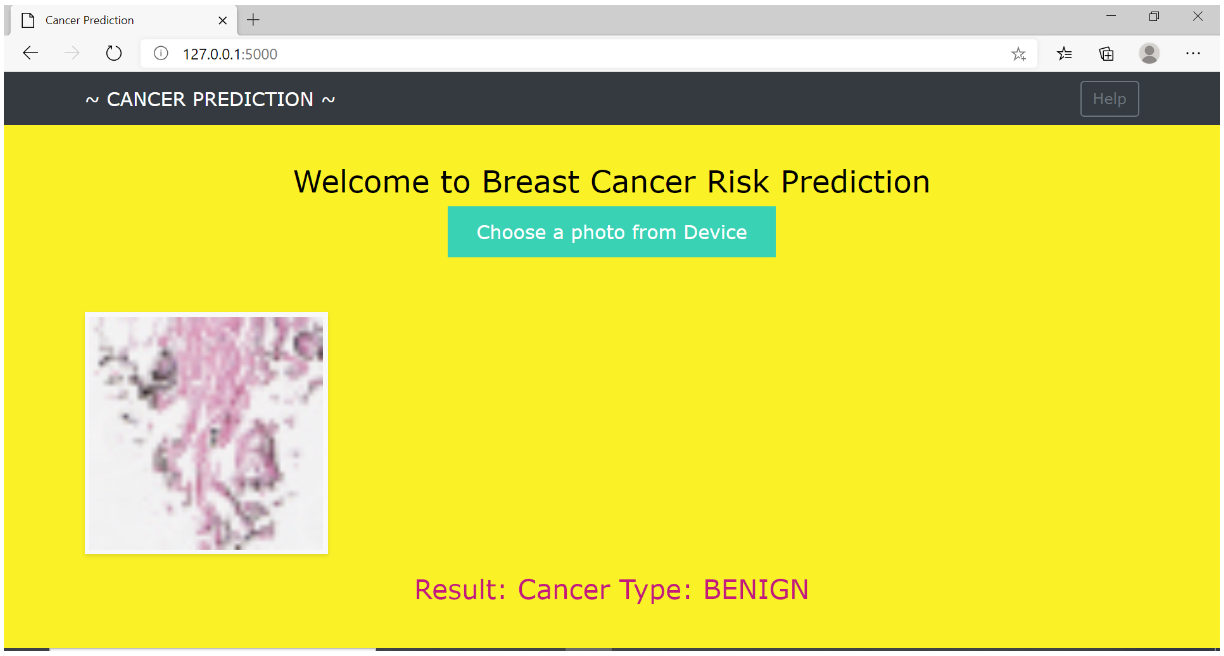
OUTPUT SCREENSHOTS :



```
(base) C:\Users\Sreeja\Documents\breast-cancer-classification\breast-cancer-classification>cd cancernet

(base) C:\Users\Sreeja\Documents\breast-cancer-classification\breast-cancer-classification\cancernet>py cancernet.py
Using TensorFlow backend.

(base) C:\Users\Sreeja\Documents\breast-cancer-classification\breast-cancer-classification\cancernet>cd ..

(base) C:\Users\Sreeja\Documents\breast-cancer-classification\breast-cancer-classification>py train_model.py
Using TensorFlow backend.
Found 20751 images belonging to 2 classes.
Found 2305 images belonging to 2 classes.
Found 5764 images belonging to 2 classes.
WARNING:tensorflow:From C:\Users\Sreeja\Anaconda3\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py:435: colocate_with (from tensorflow.python.framework.ops)
 is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
2020-07-13 21:23:26.988492: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX
2
2020-07-13 21:23:26.995421: I tensorflow/core/common_runtime/process_util.cc:71] Creating new thread pool with default inter op setting: 8. Tune using inter_op_parallelism_
threads for best performance.
WARNING:tensorflow:From C:\Users\Sreeja\Anaconda3\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and
 will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/10
648/648 [==============================] - 178s 275ms/step - loss: 0.3398 - accuracy: 0.8636 - val_loss: 1.8256 - val_accuracy: 0.8481
Epoch 2/10
648/648 [==============================] - 178s 274ms/step - loss: 0.2943 - accuracy: 0.8780 - val_loss: 0.5101 - val_accuracy: 0.8777
Epoch 3/10
648/648 [==============================] - 176s 271ms/step - loss: 0.2859 - accuracy: 0.8806 - val_loss: 1.0930 - val_accuracy: 0.8803
Epoch 4/10
648/648 [==============================] - 175s 270ms/step - loss: 0.2805 - accuracy: 0.8844 - val_loss: 0.5885 - val_accuracy: 0.8817
Epoch 5/10
648/648 [==============================] - 176s 271ms/step - loss: 0.2779 - accuracy: 0.8844 - val_loss: 0.4644 - val_accuracy: 0.8676
Epoch 6/10
648/648 [==============================] - 170s 262ms/step - loss: 0.2777 - accuracy: 0.8834 - val_loss: 1.7873 - val_accuracy: 0.8834
Epoch 7/10
648/648 [==============================] - 169s 261ms/step - loss: 0.2732 - accuracy: 0.8873 - val_loss: 1.3313 - val_accuracy: 0.8905
Epoch 8/10
648/648 [==============================] - 170s 262ms/step - loss: 0.2701 - accuracy: 0.8915 - val_loss: 0.8534 - val_accuracy: 0.8962
```

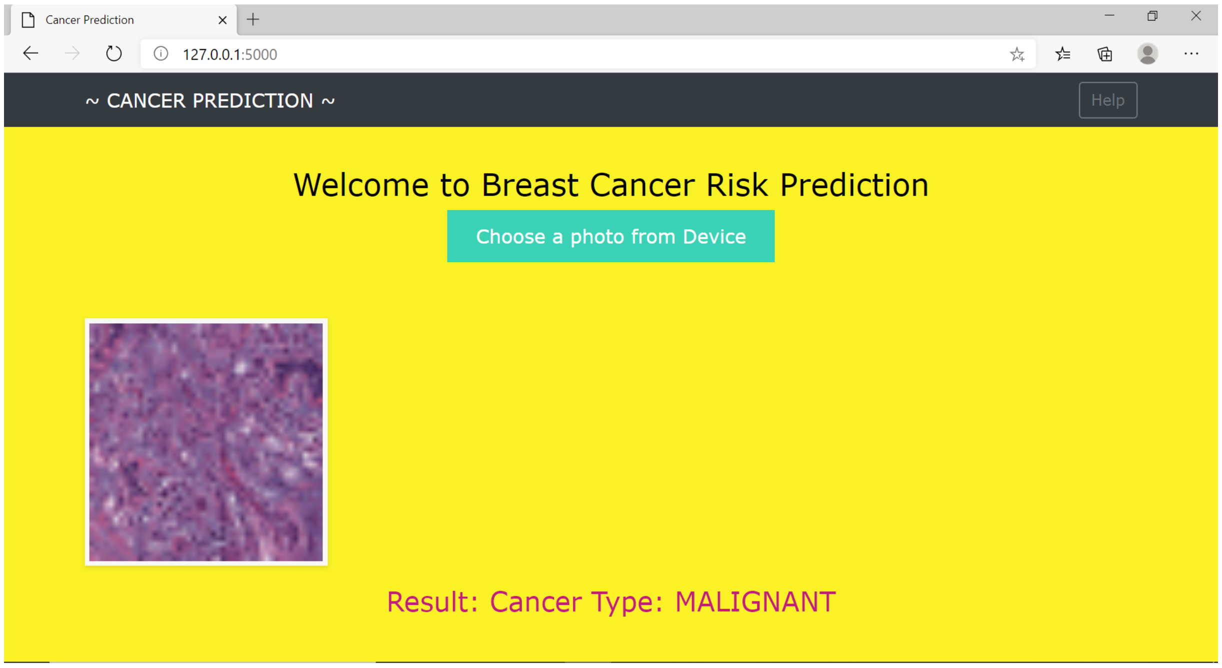


```
Epoch 4/10
648/648 [==============================] - 175s 270ms/step - loss: 0.2805 - accuracy: 0.8844 - val_loss: 0.5885 - val_accuracy: 0.8817
Epoch 5/10
648/648 [==============================] - 176s 271ms/step - loss: 0.2779 - accuracy: 0.8844 - val_loss: 0.4644 - val_accuracy: 0.8676
Epoch 6/10
648/648 [==============================] - 170s 262ms/step - loss: 0.2777 - accuracy: 0.8834 - val_loss: 1.7873 - val_accuracy: 0.8834
Epoch 7/10
648/648 [==============================] - 169s 261ms/step - loss: 0.2732 - accuracy: 0.8873 - val_loss: 1.3313 - val_accuracy: 0.8905
Epoch 8/10
648/648 [==============================] - 170s 262ms/step - loss: 0.2701 - accuracy: 0.8915 - val_loss: 0.8534 - val_accuracy: 0.8962
Epoch 9/10
648/648 [==============================] - 170s 262ms/step - loss: 0.2675 - accuracy: 0.8919 - val_loss: 0.8349 - val_accuracy: 0.8883
Epoch 10/10
648/648 [==============================] - 170s 262ms/step - loss: 0.2652 - accuracy: 0.8908 - val_loss: 1.4440 - val_accuracy: 0.8931
Now evaluating the model
              precision    recall  f1-score   support

           0       0.93      0.93      0.93      4662
           1       0.70      0.70      0.70      1102

   micro avg       0.89      0.89      0.89      5764
   macro avg       0.82      0.81      0.81      5764
weighted avg       0.89      0.89      0.89      5764

[[4337  325]
 [ 335  767]]
Accuracy: 0.8854961832061069
Specificity: 0.6960072595281307
Sensitivity: 0.9302874302874303
Traceback (most recent call last):
  File "train_model.py", line 85, in <module>
    plt.plot(np.arange(0,N), M.history["acc"], label="train_acc")
KeyError: 'acc'

(base) C:\Users\Sreeja\Documents\breast-cancer-classification\breast-cancer-classification>
```



Welcome to Breast Cancer Risk Prediction

Result: Cancer Type: BENIGN



Welcome to Breast Cancer Risk Prediction

Result: Cancer Type: MALIGNANT

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